TWO METHODS FOR THE NUMERICAL CALCULATION OF ACOUSTIC NORMAL MODES IN THE OCEAN

Kirk Eden Evans

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

TWO METHODS FOR THE
NUMERICAL CALCULATION
OF
ACOUSTIC NORMAL MODES IN THE OCEAN

by

Kirk Eden Evans

Thesis Advisors G.H. Jung & A.B. Coppens

September 1973

Two Methods for the Numerical Calculation of Acoustic Normal Modes in the Ocean

by

Kirk Eden Evans
Lieutenant, United States Navy
B.A., Miami University, 1966

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OCEANOGRAPHY

from the

NAVAL POSTGRADUATE SCHOOL
September 1973

ABSTRACT

Three computer programs were written to find the eigenvalues and eigenfunctions of acoustic normal modes in the ocean. The programs used two different methods: an iterative finite difference scheme, and a method based upon the WKB approximation of quantum mechanics. The methods assume a flat fluid bottom and are designed for any arbitrary sound speed profile. While the results of both the finite difference and the WKB methods agreed, the WKB method proved faster.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

TABLE OF SYMBOLS

The following symbols are used within the text of this thesis, but do not necessarily apply to the computer programs. The figures in parentheses refer to the equation which defines the symbol, or in which the symbol is first mentioned.

a - coefficient for particular solution of Z_1 (57).

Ai - Airy phase solution.

b - coefficient for particular solution of Z_1 (57).

Bi - Airy phase solution.

c - sound velocity.

c_b - bottom sound velocity.

c_{min} - minimum sound velocity in the sound velocity profile.

C - coefficient for particular solution of Z_2 (24).

C_g - group speed (40).

C_p - phase speed (39).

D - coefficient for particular solution of Z_2 (24).

e - 2.718281828459045.

E - energy level, or horizontal wave function (17).

$F(z)$ - square of the vertical wavenumber (92).

h - depth grid spacing.

H - depth from upper (lower) turning point to surface (bottom) (102).

$]-[o^{(1,2)}$ Hankel functions of the first and second kind.

k - wave number (14).

k_{max} - maximum possible real wave number (16).

k_r - horizontal wave number (14).

K_{r_m} - eigenvalue (horizontal wave number).

k_z - vertical wave number (14).

K_B - bottom imaginary vertical wave number (24).

K_s - imaginary vertical wave number at water side of bottom (107).

K_z - imaginary vertical wave number (49).

L - integral of imaginary wave number across a barrier (86).

m - mode number.

N - maximum number of discrete modes available (33).

p - acoustic pressure.

r - horizontal range.

R - horizontal displacement potential (10).

R_1 - reflection coefficient for a barrier (85).

$S(z)$ - an integral function of depth (34).

$S_1(z)$ - vertical wave number integral for Z_1 (65).

$S_2(z)$ - vertical wave number integral for Z_2 (64).

t - time.

$V(z)$ - velocity function (16).

V_b - bottom value of the velocity function.

z - depth.

z_b - bottom depth.

z_{TL} - lower turning point depth (64).

z_{TU} - upper turning point depth (66).

z_1 - top depth of a barrier (86).

z_2 - bottom depth of a barrier (86).

Z - depth dependent vertical displacement potential (13).

Z_b - bottom vertical displacement potential (22).

Z_o - vertical displacement potential at water side of bottom boundary.

Z_m - eigenfunction for a particular mode, m.

Z_1 - WKB solution in the "real" region (57).

Z_2 - WKB solution in the "imaginary" region (59).

α - mode attenuation coefficient (45).

β - bottom volume attenuation coefficient (45).

ΔS - difference function (80).

$\zeta(z)$ - an undefined function of depth (34).

θ_l - phase at lower turning point (63).

θ_u - phase at upper turning point (67).

θ_1 - phase at top of barrier (115).

θ_2 - phase at bottom of barrier (115).

K_B - complex wave number in the bottom (43).

K_r - complex horizontal wave number (44).

ν - mode normalization integral (41).

ξ - an undefined function of depth (35).

π - 3.141592653589793

ρ - density.

ρ_b - bottom density.

ρ_o - ocean water density.

σ - velocity weighted mode normalization integral (42).

ϕ - angle made by ray with the horizontal (grazing).

Φ - displacement potential (2).

ω - angular frequency.

I. INTRODUCTION

Since the Second World War considerable interest has developed, in both naval and scientific communities, in the prediction of sound propagation within the ocean. For the scientist, propagation information can be a vital part in investigating the acoustic, physical and sometimes chemical properties of the ocean. For the naval community, propagation is a vital element in the prediction of acoustic sensor performance. The prediction of such performance is not limited to the design and development of sensor systems, but is increasingly important in the operational employment of such sensors and selection of the most fruitful tactics.

Accordingly, there has been considerable development within the naval community of numerical acoustic propagation models. These models have, for the most part, been based upon the theory of ray acoustics. The techniques employed have developed considerable sophistication in order to deal with ray limitations with caustics, frequency effects, and interference. However, for prediction of propagation over great ranges and at low frequencies, ray techniques have two serious limitations:

(1) The long ranges involved require long computation times. Each ray must be "traced" over many successive short time steps to simulate travel of the wavefront. Usually more than one hundred such rays are traced. Thus, as ranges increase, the computation time

increases proportionately. Such long computation times make the routine operational use of ray trace programs covering paths of thousands of kilometers too expensive to be practical.

(2) Ray trace programs largely ignore such frequency dependent effects as diffraction and interference. The assumptions of ray acoustics require that the wavelength of the acoustic signal be short in comparison to the depth and velocity gradient scale. As lower frequency signals are considered, the wavelength becomes an appreciable fraction of the depth scale, and so violates a basic validity assumption for ray acoustics.

Because of these limitations there has been increased interest in normal mode theory as a basis for long range propagation models. The theory of normal mode propagation is not new and has had a number of shallow water applications [Officer 1958, Bucker and Morris 1965]. A. O. Williams (1970) has outlined a method whereby the normal mode technique could be applied to deep ocean acoustics. As a part of any such technique, both the characteristic horizontal wavenumber (eigenvalue) and the mode shape (eigenfunction) must be solved for each mode. To do this, three basic methods have been used:

(1) The wave equation has been solved in closed form [Tolstoy and Clay 1966, Williams and Horne 1967]. In order to do this, the sound velocity profile must be fitted to an assumed analytic function. Such functions have taken the form of Epstein profiles [Bucker and

Morris 1967], profiles with a constant gradient of the reciprocal of the sound velocity or sound velocity squared [Williams and Horne 1967, Tolstoy and Clay 1966]. The requirement that the sound velocity profile be fitted to an arbitrary function places severe limitations on the flexibility of any model.

(2) The wave equation has been vertically integrated while meeting appropriate boundary conditions. This method has the advantage of being adaptable to any arbitrary sound velocity profile. Kanabis (1972) and Newman and Ingenito (1972) have developed two such shallow water normal mode models. The first program presented with this thesis, NORMO1, is based upon these two shallow water models.

(3) The Wentzel-Kramers-Brillouin (WKB) approximation of quantum mechanics provides an analytic solution to the wave equation. This solution is singular at depths which correspond to ray vertex depths. However, the WKB approximation may lend itself to solution of the eigenvalue, after which the eigenfunction may be solved by an integration similar to the previous method. The last two programs presented with this thesis, NORMO4 and NORMO5, use this method.

The long term goal of this research is to develop a deep ocean normal mode acoustic propagation model of sufficient computational speed to be considered for operational use. In order to achieve this goal, a rapid method of solving for the eigenfunction must first be developed. Based upon this need the immediate objectives of this thesis are:

(1) To compare the results of the programs employing methods (2) and (3) above with analytic solutions, the published results of other programs, and each other.

(2) Determine whether the WKB method offers promise of improvement in terms of time versus accuracy.

(3) To outline future improvements to either method based upon the preliminary results.

II. THEORY

A. WAVE THEORY

In this section we apply a mathematical separation of variables to the wave equation; then we impose boundary conditions on the resulting separated vertical equation. The results of this mathematical procedure offer us a differential equation and boundary conditions capable of numerical solution. We will close this section with a discussion of some general properties of such a solution.

The following discussion is based upon the treatments given by A. O. Williams (1970), and I. Tolstoy and C. S. Clay (1966). For a more complete descriptive treatment the reader is referred to the discussion given by Williams.

1. The Wave Equation

The simple scalar wave equation for small amplitude waves is

$$\nabla^2 \Phi = \frac{1}{C^2} \frac{\partial^2 \Phi}{\partial t^2} , \tag{1}$$

where Φ is the displacement potential. Here $\vec{d}$, the displacement of a fluid particle from its rest position, is represented by

$$\vec{d} = \nabla \Phi , \tag{2}$$

and the acoustic pressure, p, is given by

$$p = -\rho \frac{\partial^2 \Phi}{\partial t^2} = -\rho c^2 \nabla^2 \Phi \,. \tag{3}$$

Thus, if we can express Φ as a function of space, we can then find the resulting acoustic pressure and sound propagation loss due to spreading and refraction.

If we use a cylindrical coordinate system and assume Φ is a function of range, depth, and time, we can rewrite (1) as

$$\frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial \Phi}{\partial r} \right) + \frac{\partial^2 \Phi}{\partial z^2} - \frac{1}{c^2} \frac{\partial^2 \Phi}{\partial t^2} = 0. \tag{4}$$

Let us further assume that Φ is separable in terms of r, z, and t, such that

$$\Phi(r,z,t) = R(r)\, Z(z)\, e^{i\omega t}, \tag{5}$$

where ω is the source angular frequency, Z(z) is the vertical displacement potential function, and R(r) is the radial displacement potential function. We immediately notice that

$$\frac{\partial^2 \Phi}{\partial t^2} = -\omega^2 \Phi \,. \tag{6}$$

Using equations (5) and (6) we rewrite equation (4) as

$$\frac{1}{r R} \frac{\partial}{\partial r} \left(r \frac{\partial R}{\partial r} \right) + \frac{1}{Z} \frac{\partial^2 Z}{\partial z^2} + \frac{\omega^2}{c^2} = 0. \tag{7}$$

This expression has a range dependent term, a depth dependent term and a term with c, the sound velocity. If we could assume that c is independent of either depth or range, the expression is separable into depth and range differential equations. The obvious choice is to assume that c varies only with depth. This assumption, termed horizontal stratification, although not true over great distance, is commonly made in oceanography and ocean acoustics. Actually, for our purposes it will be sufficient if c is horizontally stratified in the vicinity of our solution, and further if any horizontal variation of c is small with respect to the vertical variation. Making such an assumption, we continue by separating (7) into range and depth differential equations

$$\frac{1}{rR}\frac{\partial}{\partial r}\left(r\frac{\partial R}{\partial r}\right) = -k_r^2 , \tag{8}$$

$$\frac{1}{Z}\frac{\partial^2 Z}{\partial z^2} + \frac{\omega^2}{c^2} = k_r^2 . \tag{9}$$

Solutions to equation (8) include the Hankel functions of first and second kind representing cylindrical waves,

$$R(r) = \text{]-[}_0^{(1,2)}(k_r r) . \tag{10}$$

For distances such that $k_r r$ is much greater than one, this function can be approximated by its asymptotic form

$$R(r) \cong \sqrt{\frac{2}{\pi k_r r}}\, e^{\pm i\left(k_r r - \frac{\pi}{4}\right)} . \tag{11}$$

We now have an assumed time dependence and a solution to the range dependent function of Φ. Thus, at long range from the source we have for Φ

$$\Phi = Z(z)\sqrt{\frac{2}{\pi k_r r}}\; e^{-i\left(k_r r - \frac{\pi}{4} - \omega t\right)} . \tag{12}$$

Here k_r^2, an eigenvalue which appeared in the separation process as a mathematical constant, will be seen to be a horizontal wave number. We will consider only positive values of k_r, which represent outgoing wavefronts. It remains for us to find a method of solving for $Z(z)$ in terms of the separation parameter, k_r^2.

We can rewrite (9) as

$$\frac{\partial^2 Z}{\partial z^2} - \left(\frac{\omega^2}{c^2} - k_r^2\right) Z = 0 . \tag{13}$$

This equation is the separated, space form of the wave equation, or Helmholtz equation, for the vertical wave component. The expression in parentheses represents a vertical wave number, k_z, and is related to the (true) wave number, k, and horizontal wave number, k_r, by

$$k^2 = \frac{\omega^2}{c^2} = k_z^2 + k_r^2 . \tag{14}$$

Equation (13) with a pair of associated homogeneous boundary conditions forms a Sturm-Liouville problem, which is solved in terms of a set of eigenvalues and corresponding eigenfunctions. It now remains for us to define and employ boundary conditions associated with the ocean vertical sound profile. Before so doing, however, we will digress a moment to introduce a notational convenience.

Since the value of c does not vary greatly (perhaps five percent) throughout the depth profile, the square of the vertical wave number represents a small difference between two nearly equal numbers

$$k_z^2 = \frac{\omega^2}{c^2} - k_r^2 \ . \tag{15}$$

In order to conserve accuracy and facilitate computational speed, a notational convention is borrowed from quantum mechanics. The square of the (true) wave number, k^2, is subtracted from an arbitrary constant of the same order of magnitude (here the maximum possible wave number) to form a "potential function."

$$V(z) = \frac{\omega^2}{c_{min}^2} - \frac{\omega^2}{c^2} = k_{max}^2 - \frac{\omega^2}{c^2} \ . \tag{16}$$

The square of the horizontal wave number (our separation constant) is also subtracted from the maximum wave number to form a quantity corresponding to the "energy level" of quantum mechanics, and our new eigenvalue

$$E = k_{max}^2 - k_r^2 \ . \tag{17}$$

With this notation, the Helmholtz equation (13) becomes

$$\frac{\partial^2 Z}{\partial z^2} - \left[E - V(z)\right] Z = 0 \ , \tag{18}$$

a form similar to Schroedinger's equation. By this notation we hope to facilitate the adaptation of the results of quantum mechanics to the problem at hand.

2. Boundary Conditions

We continue by developing appropriate vertical boundary conditions.

Figure 1 represents a typical deep ocean sound profile. Figure 2 is the same profile represented in terms of V(z). Note that at some intermediate depth the value of sound velocity reaches a minimum at the sound channel axis. Note also that the function V(z) forms a "potential well" with depth, representing the deep sound channel. Our domain of interest is bounded by the sea surface and a flat bottom.

The air-water boundary at the surface approximates a free surface at which the stresses vanish. In a fluid this boundary corresponds to

$$Z(o) = 0. \tag{19}$$

If the limit of the second derivative of the vertical displacement function is finite at the surface, the stipulation that V(z) discontinuously approaches infinity at the surface has the effect of making Z(z) vanish at the surface. Thus, in terms of our quantum mechanics analogy, the free surface boundary condition corresponds to a perfectly rigid wall of the potential well.

The ocean bottom provides a boundary more difficult to specify. The ocean bottom has a complex, often layered, horizontally varying structure. In addition, little is known of the specific

TYPICAL DEEP OCEAN SOUND PROFILE

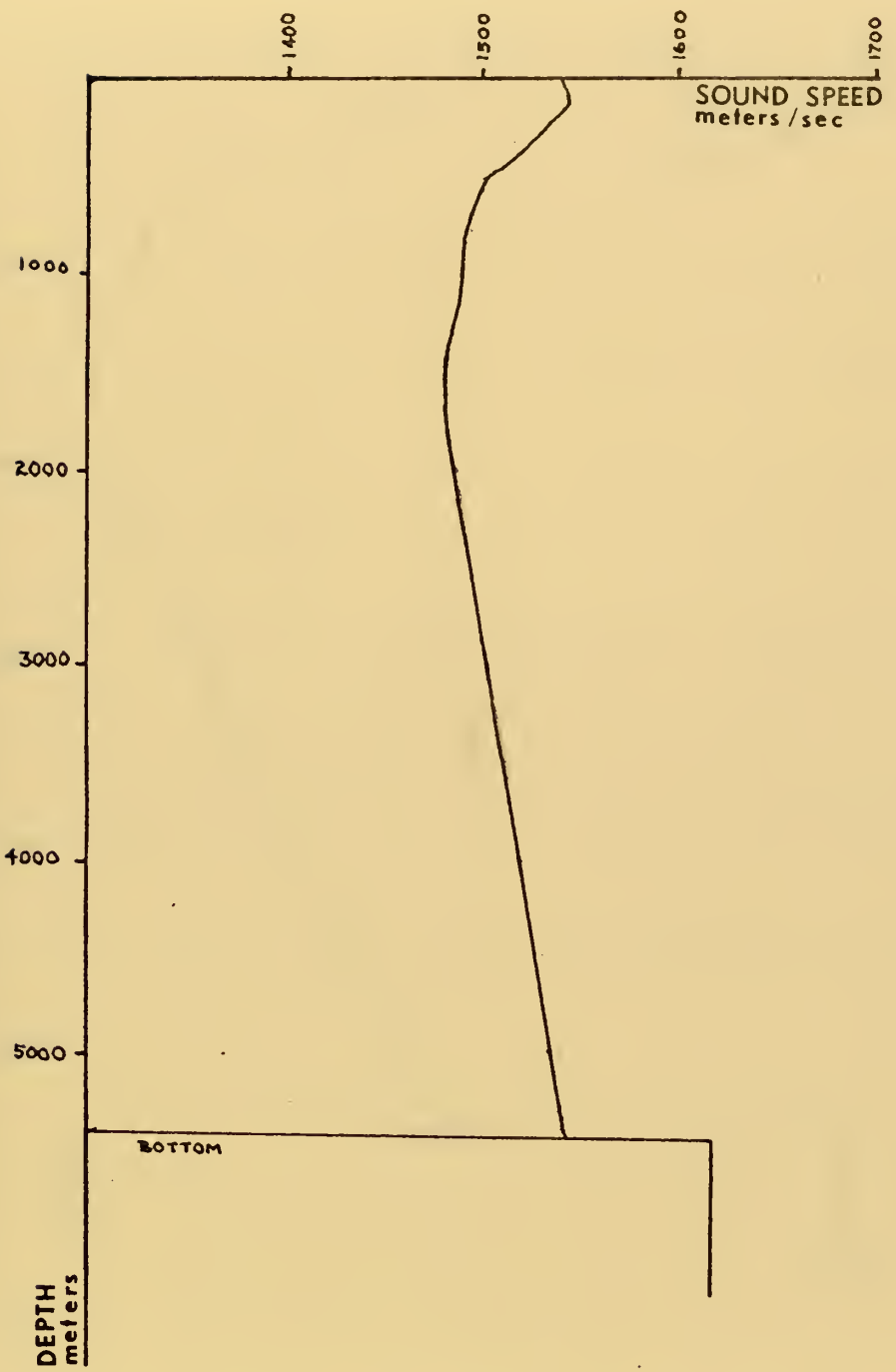


Figure 1

TYPICAL DEEP OCEAN VELOCITY FUNCTION

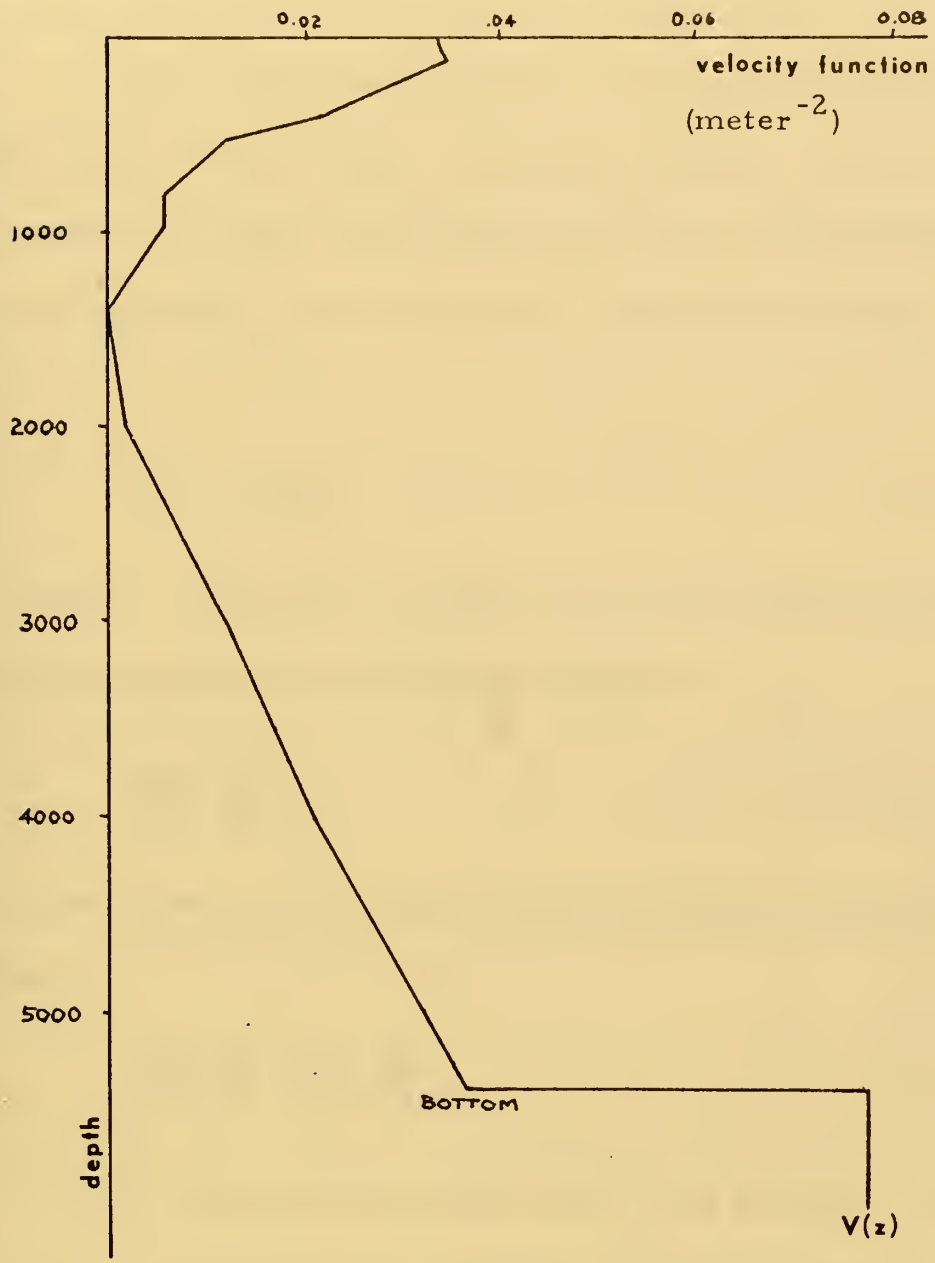


Figure 2

acoustic properties of much of the bottom required to model fully the bottom boundary conditions. In the face of such complexities we will approximate the bottom with a relatively simple model.

We will assume that the bottom is represented by a homogeneous fluid of infinite depth, and constant velocity. Across a fluid-fluid interface we require that both pressure and vertical particle motion be continuous. From (3) and (6), continuity of pressure requires

$$\rho_o Z_o = \rho_b Z_b \tag{20}$$

at the bottom. From (2) we find that continuity of vertical particle motion or displacement at the bottom requires

$$\frac{\partial Z_o}{\partial z} = \frac{\partial Z_b}{\partial z} \quad . \tag{21}$$

We combine equations (20) and (21) to form a single bottom boundary condition

$$\frac{1}{\rho_o Z_o}\frac{\partial Z_o}{\partial z} = \frac{1}{\rho_b Z_b}\frac{\partial Z_b}{\partial z} \quad . \tag{22}$$

Since we have specified the bottom to be of constant velocity and density, a known solution to equation (18) for the bottom region is

$$Z_b = c_1 e^{K_B z} + c_2 e^{-K_B z} \quad , \tag{23}$$

where K_B, a real constant, is the imaginary bottom wave number,

$$K_B = \sqrt{V_B - E} \quad . \tag{24}$$

If the amount of energy represented by the bottom solution is to be finite, the limit of the integral of Z_b from the bottom boundary to infinity must in turn be finite. This condition requires that the exponential growth term of Z_b be suppressed. Thus, (23) becomes

$$Z_b = c_2 e^{-K_B z} \quad , \tag{25}$$

and (22) becomes

$$\frac{1}{Z_o}\frac{\partial Z_o}{\partial z} = -\frac{\rho_o}{\rho_b}\sqrt{V_B - E} \quad . \tag{26}$$

We have now specified two boundary conditions which will enable us to find particular solutions to the vertical differential wave equation (18). Before proceeding with the technique of finding such solutions, let us discuss some of their properties.

3. The Nature of the Solution

Before we find the solutions, $Z(z)$, to the differential equation (18) and boundary conditions (equations 19 and 26), it will be beneficial to consider the form and properties of the solutions we expect.

a. In the previous section we saw that the solution in the bottom region is of the form of a decaying exponential (equation 25). This solution requires that the quantity K_B in equation 25 must be real. This requirement in turn implies that

$$E < V_b \tag{27}$$

or

$$k_r > \frac{\omega}{c}_b . \tag{28}$$

In order to have propagating waves, it must also hold that

$$E > 0, \tag{29}$$

or

$$k_r < \frac{\omega}{c}_{min} . \tag{30}$$

Thus, we have an upper and lower bound for E and k_r,

$$V_b > E > 0 \tag{31}$$

and

$$\frac{\omega}{c}_{min} > k_r > \frac{\omega}{c}_b . \tag{32}$$

b. Within these limits the boundary value problem is so constrained that it can only be solved in terms of a finite number of eigenvalues, E_m, and corresponding functions $Z_m(z)$. This eigenfunction, properly normalized, forms what is termed a normal mode. At sufficient range a vertical sound pressure profile can be approximated by a linear combination of such eigenfunctions,

$$p(z) \approx \sum_{m=1}^{N} c_m Z_m(z) . \tag{33}$$

TYPICAL MODE SOLUTION

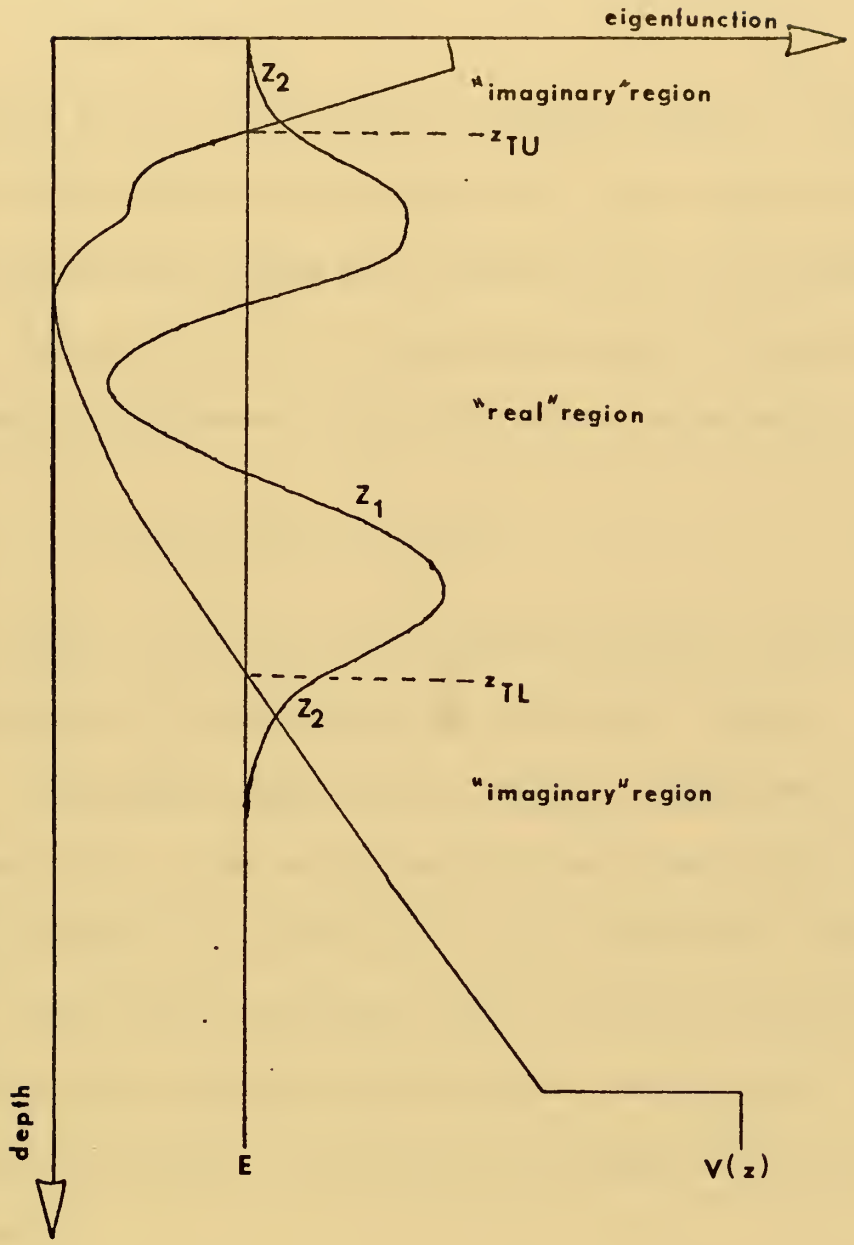


Figure 3

c. At depths such that $E_m > V(z)$ the eigenfunction $Z_m(z)$ is oscillatory in character. We can represent such a function by

$$Z(z) = \xi(z) \sin(S(z)) \tag{34}$$

where $\xi(z)$ and $S(z)$ are undefined functions of depth. We shall refer to such depths as the oscillatory or "real" region (a reference to the fact that the vertical wave number, k_z, is a real number). At depths such that $E_m < V(z)$ the eigenfunction $Z_m(z)$ becomes quasi-exponential in character, with a form we can represent by

$$Z(z) = \xi(z) [C e^{S(z)} + D e^{-S(z)}] \tag{35}$$

where, again, $\xi(z)$ and $S(z)$ are undefined functions. We shall refer to such depths as the exponential or "imaginary" region.

The depth at which the sign of $[E_m - V(z)]$ changes is termed a turning point. At the turning point the character of the eigenfunction changes from oscillatory to quasi-exponential (Fig. 3).

d. Each mode corresponds to one or more rays which traces paths within the oscillatory region between turning points, surface, or bottom boundaries. The angle the ray makes with the horizontal is defined by

$$\cos \phi_m(z) = c(z) \frac{k_r}{\omega} . \tag{36}$$

From this we see that the mode turning point corresponds to the ray vertex depth (that depth at which a ray reaches its maximum depth excursion). When the oscillatory region of a mode is bounded by the

free surface boundary or the bottom, the corresponding ray is surface or bottom reflected, respectively.

e. The lower boundary placed on k_r represented by equation (28) has a more familiar and perhaps more satisfying physical meaning. Substituting equation (36) into (28) we have

$$\text{Cos } \phi > \frac{c(z)}{c_b} . \tag{37}$$

At the bottom this is the expression for the critical angle. Thus, we are limiting outselves to consider only those modes whose corresponding ray strikes the bottom at a grazing angle less than the critical angle. Such modes are widely called "unattenuated modes." Also, there exists an infinite set of modes such that

$$k_r < \frac{\omega}{c_b} \tag{38}$$

However, because of bottom reflection loss, those modes tend to attenuate rapidly with range. For propagation problems at a considerable range, the "attenuated modes" contribute an insignificant amount to the acoustic pressure field, and are ignored.

f. Given the oscillatory nature of the eigenfunction Z_m, we see that the eigenfunctions for each successive mode must change sign one additional time (Figure 4). Each sign change will be referred to as a mode crossing. Thus, corresponding to each eigenvalue E_m, there corresponds an eigenfunction Z_m with m-1 mode crossings.

SUCCESSIVE MODES

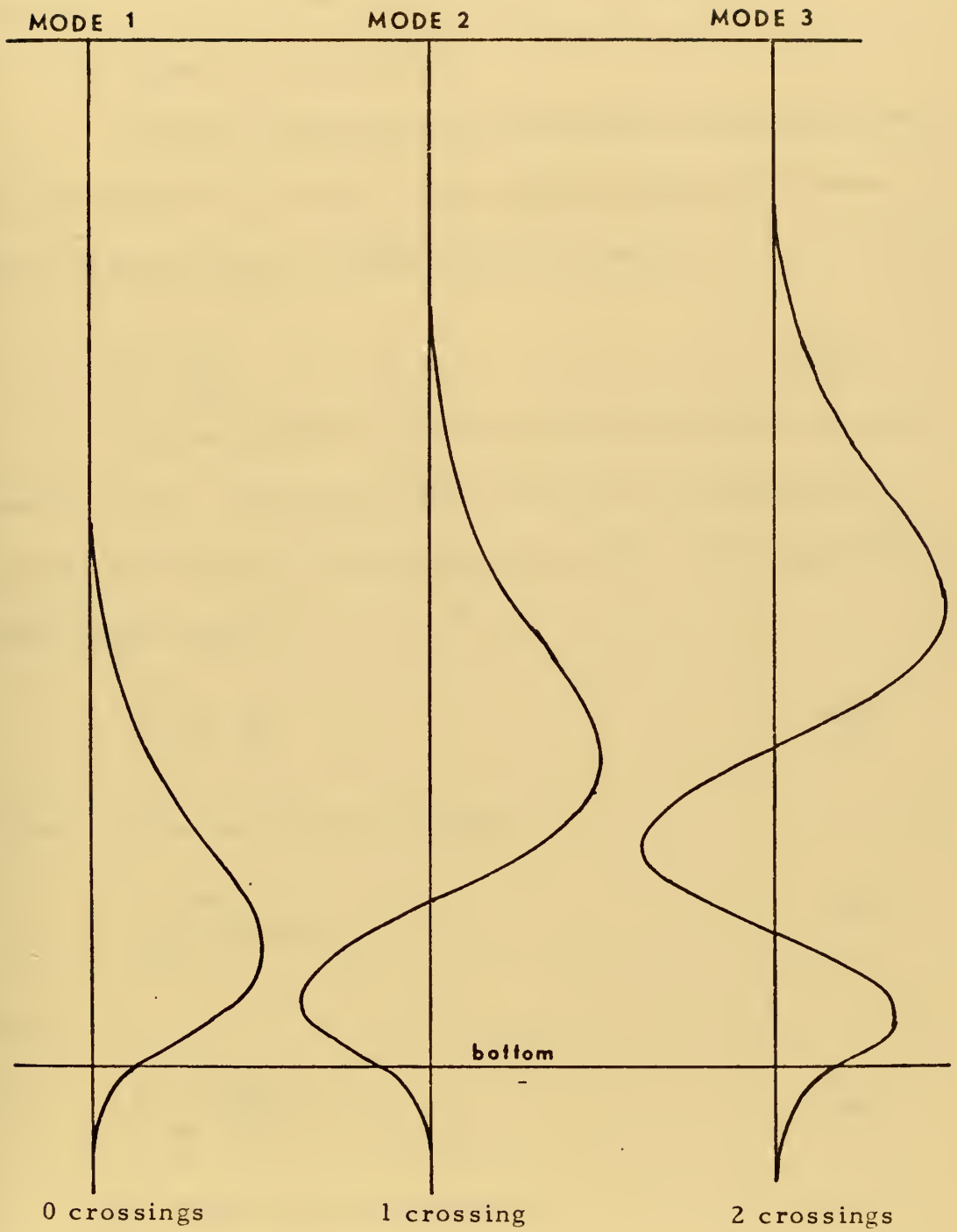


Figure 4

4. Mode Parameters

We should now introduce two sets of parameters, derived elsewhere, which are used in normal mode calculations.

a. Phase and Group Speed

Phase speed describes the horizontal speed of the wavefront represented by a mode. As the speed of advance of a wavefront of constant phase, the phase speed is given by

$$C_p = \frac{\omega}{k_r} . \tag{39}$$

Group speed is the rate of energy transport in the horizontal direction. Tolstoy and Clay (1966) give an expression for group speed based upon a theorem by Biot (1957). This method calculates group speed by

$$C_g = \frac{1}{C_p} \frac{\nu}{\sigma} \tag{40}$$

where ν is the normalization integral,

$$\nu = \int_{-\infty}^{\infty} \rho Z(z)^2 \, dz , \tag{41}$$

and

$$\sigma = \int_{-\infty}^{\infty} \rho \frac{Z^2(z)}{C(z)^2} \, dz . \tag{42}$$

b. Bottom Decay Coefficient

Kornhauser and Raney (1955) give an expression which calculates the effect of a bottom absorption on the mode amplitude. Assume the wave number in the bottom is complex and given by

$$\underline{K}_b = \frac{\omega}{c_b} + i\beta \tag{43}$$

where β is a bottom attenuation coefficient. Then the horizontal wave number must also be complex and given by

$$\underline{K}_r = k_r + i\alpha . \tag{44}$$

If β is small with respect to $\frac{\omega}{c_b}$, then the ratio of α to β for a given mode is approximated by

$$\frac{\alpha}{\beta} = \frac{\omega \rho_b}{c_b k_r V} \int_{z=\text{bottom}}^{\infty} Z^2(z)\, dz . \tag{45}$$

B. THE WENTZEL-KRAMERS-BRILLOUIN (WKB) APPROXIMATION

The WKB approximation is a method borrowed from quantum mechanics which will enable us to approximate the eigenvalues, E_m, of equation (18). In the WKB approximation we assume a form for the mode solution which is valid at depths removed from the turning points. We then develop a characteristic equation for the eigenvalue, based upon the assumed solution. This section follows in many respects the descriptions given by Tolstoy and Clay (1966) and Lauvstad (1971).

1. The Characteristic Equation

Based upon the sign of [E - V(z)] the vertical wave equation (18) applies to two domains, which we have termed the "imaginary" region and "real" region. Let us rephrase equation (18) into two separate equations for the two domains. So doing, we have

$$Z_1'' + k_z^2 Z_1 = 0 \qquad \text{for} \quad E > V(z), \qquad (46)$$

and

$$Z_2'' - K_z^2 Z_2 = 0 \qquad \text{for} \quad E < V(z). \qquad (47)$$

Here we have defined

$$k_z = [E - V(z)]^{1/2} \qquad \text{when} \quad E > V(z), \qquad (48)$$

and

$$K_z = [V(z) - E]^{1/2} \qquad \text{when} \quad E < V(z). \qquad (49)$$

Now let us assume a form for the solution to equation (46), similar to that which we have previously postulated in equation (34). The assumed solution is

$$Z(z) = \mathcal{S}(z)\, e^{iS(z)}. \qquad (50)$$

By substituting equation (50) into (46) we have as a real part

$$S''(z) - \mathcal{S}(z)(S'(z)^2 - k_z^2) = 0, \qquad (51)$$

and an imaginary part

$$\mathcal{S}(z) S''(z) + 2\mathcal{S}'(z) S'(z) = 0. \qquad (52)$$

This equation immediately yields an expression for $\mathcal{S}(z)$,

$$\mathcal{S}(z) = A\sqrt{S'(z)}. \qquad (53)$$

Now let us simplify equation (51) with an assumption. Assume that the first term of equation (51) is negligible with respect to the others, or

$$\frac{1}{k_z^2}\left|\frac{S''(z)}{S(z)}\right| \ll 1 \quad . \tag{54}$$

With this assumption we obtain as an expression for S(z)

$$S'(z)^2 = k_z^2 \quad , \tag{55}$$

or upon integrating

$$S_1(z) = \pm \int_{z_0}^{z} k_z \, dz \quad . \tag{56}$$

We now have as the assumed solution

$$Z_1(z) = k_z^{-1/2}\left[a \operatorname{Cos} S_1(z) + b \operatorname{Sin} S_1(z)\right], \tag{57}$$

or, in a more convenient form,

$$Z_1(z) = k_z^{-1/2}\left[a \operatorname{Sin}(S_1(z) + \theta_0)\right] . \tag{58}$$

Similarly we obtain as a corresponding solution for the "imaginary" region,

$$Z_2(z) = K_z^{-1/2}\left[Ce^{S_2(z)} + De^{-S_2(z)}\right], \tag{59}$$

where $S_2(z)$ is defined by

$$S_2(z) = \int_{z_0}^{z} K_z \, dz \quad . \tag{60}$$

Having written the two solutions, let us take a closer look at the assumption implied by equation (54). With equations (53) and (55) we can rewrite (54) for the "real" region as

$$\frac{1}{k_z}\left|\frac{d}{dz}\log k_z\right| \ll 1 \ , \tag{61}$$

and for the "imaginary" region,

$$\frac{1}{K_z}\left|\frac{d}{dz}\log K_z\right| \ll 1 . \tag{62}$$

This assumption requires that the order of magnitude of the vertical wave number, and thus the sound speed, vary slowly with respect to depth. This is generally the case in the ocean, where the total sound speed variation with depth is on the order of five percent. We also see that our solutions lose any validity as z approaches a turning point. Recalling the analogy of the turning point to the ray vertex depth, we see that as a wave approaches the turning point its orientation becomes horizontal. Near a turning point the vertical wavenumbers, k_z and K_z, approach zero and the conditions of equations (61) and (62) are no longer satisfied. Thus the solutions represented by equations (58) and (59) become singular at the turning point location.

The WKB method does not, at this point, appear satisfactory for the computation of the mode profile or eigenfunction Z_m. However, if the WKB approximation can be used to obtain an accurate estimate of the eigenvalues, E_m, a finite difference scheme can then be used to calculate the eigenfunction, Z_m.

Consider a typical V(z) profile, with upper and lower turning points, as well as bottom and pressure release boundaries (Figure 5). Remember that we have specified a derivative boundary condition at the bottom. This boundary condition determines the eigenfunction (except for a constant of multiplication) in the "imaginary" region $z_{TL} \rightarrow z_b$. At the lower turning point, z_{TL}, the "imaginary" region solution, $Z_2(z)$, corresponds to a particular "real" region solution, $Z_1(z)$. We can designate such a solution by the use of an initial phase angle, θ_L, such that

$$\frac{1}{\sqrt{K_z}}\left(C e^{S_2(z)} + D e^{-S_2(z)}\right) \rightarrow \frac{A}{\sqrt{k_z}} \operatorname{Sin}\left(S_1(z) + \theta_L\right), \qquad (63)$$

where

$$S_2(z) = \int_{BOTTOM}^{z_{TL}} K_z \, dz \,, \qquad (64)$$

and

$$S_1(z) = \int_{z_{TL}}^{z} k_z \, dz \,. \qquad (65)$$

In a similar manner we can stipulate that in order to satisfy the free surface boundary condition, $Z(0) = 0$, the "real" region solution, Z_1, must correspond to a particular "imaginary" region solution, Z_2. Again we can represent this as a phase angle, θ_s, in the "real" region solution, Z_1. Thus we have required the argument of $\operatorname{Sin}(S_1(z) + \theta_L)$ at the upper turning point (or surface) to be our specified value

A WKB PROFILE

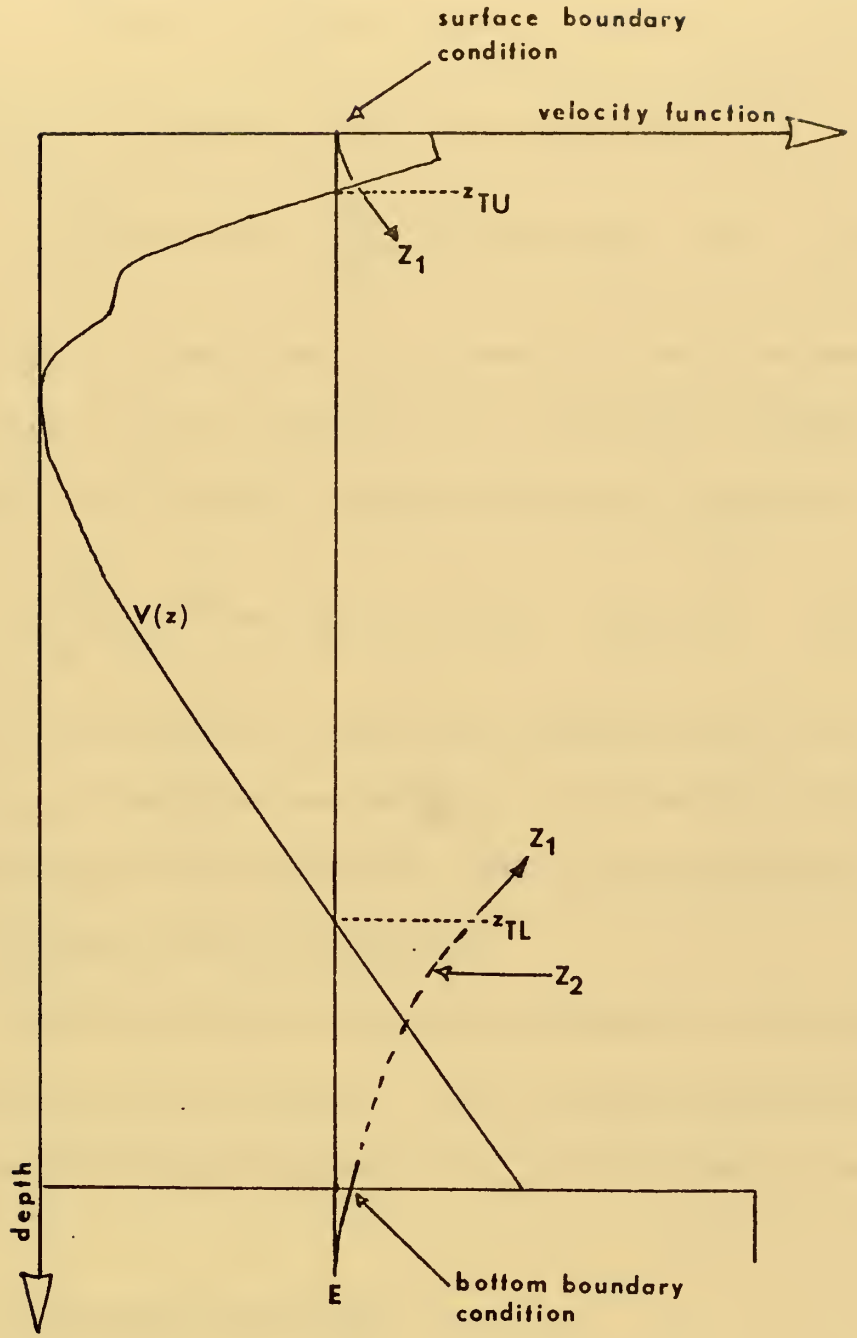


Figure 5

$$S_1(z_{TU}) + \theta_L = \int_{z_{TL}}^{z_{TU}} k_z \, dz + \theta_L = \theta_S \ . \qquad (66)$$

If we let $\theta_u = \pi - \theta_s$ we can rewrite this equation as

$$\int_{z_{TL}}^{z_{TU}} k_z \, dz + \theta_L + \theta_u = \int_{z_{TL}}^{z_{TU}} \left[E - V(z)\right]^{1/2} dz + \theta_L + \theta_u = \pi. \qquad (67)$$

Since this equation describes the arguments for which the sine of the integrated vertical wave number is zero (a Dirichlet boundary condition), the surface boundary condition will also be satisfied if

$$\int_{z_{TL}}^{z_{TU}} \left[E - V(z)\right]^{1/2} dz + \theta_L + \theta_u = m\pi \qquad m = 1, 2, \ldots, N. \qquad (68)$$

Values of E satisfying this characteristic equation will be the eigenvalues, E_m, for which we are searching. Note that this equation is also that given by Tolstoy and Clay (1966) as the characteristic equation for stratified acoustic waveguides.

Now that we have a characteristic equation, some method of evaluating θ_L and θ_u is required. The WKB approximation provides a method for finding the eigenvalues, E_m, once θ_u and θ_L are evaluated.

2. Turning Point Connection

The most vexing problem in using the WKB approximation is connecting the two solutions (58) and (59) across the turning points. A number of techniques have been developed, and here we will consider two such techniques which we will term the asymptotic and Airy phase methods.

a. Asymptotic Method

We have two corresponding solutions represented by equations (58) and (59) which correspond to the same function on either side of a turning point. As the turning point is approached let us require that the limit of the first derivative divided by the mode value be continuous. This is in fact the same as the fluid-fluid boundary condition of equation (22). For the purpose of these turning point discussions we will designate the turning point depth, z_{TP}, equal to zero and corresponding to z_0 of equations (56) and (60). Further, z will be positive towards the "real" region and negative towards the "imaginary" region (Figure 6). Thus, from equation (22) we have

$$\lim_{z \to 0_+} \frac{Z_1'}{Z_1} = \lim_{z \to 0_-} \frac{Z_2'}{Z_2} . \tag{69}$$

By taking advantage of the approximation involved in equation (61), we get by substitution for the above

$$\lim_{z \to 0_+} k_z \cot(S_{1(z)} + \theta_o) = \lim_{z \to 0_-} K_z \frac{Ce^{S_{2(z)}} - De^{-S_{2(z)}}}{Ce^{S_{2(z)}} + De^{-S_{2(z)}}} . \tag{70}$$

Taking the limit we have

$$\frac{b}{a} = \cot(\theta_o) = \frac{D - C}{D + C} . \tag{71}$$

It should be noted that this approach is simpler than the Airy phase technique (described next) and may not be as accurate. It is included,

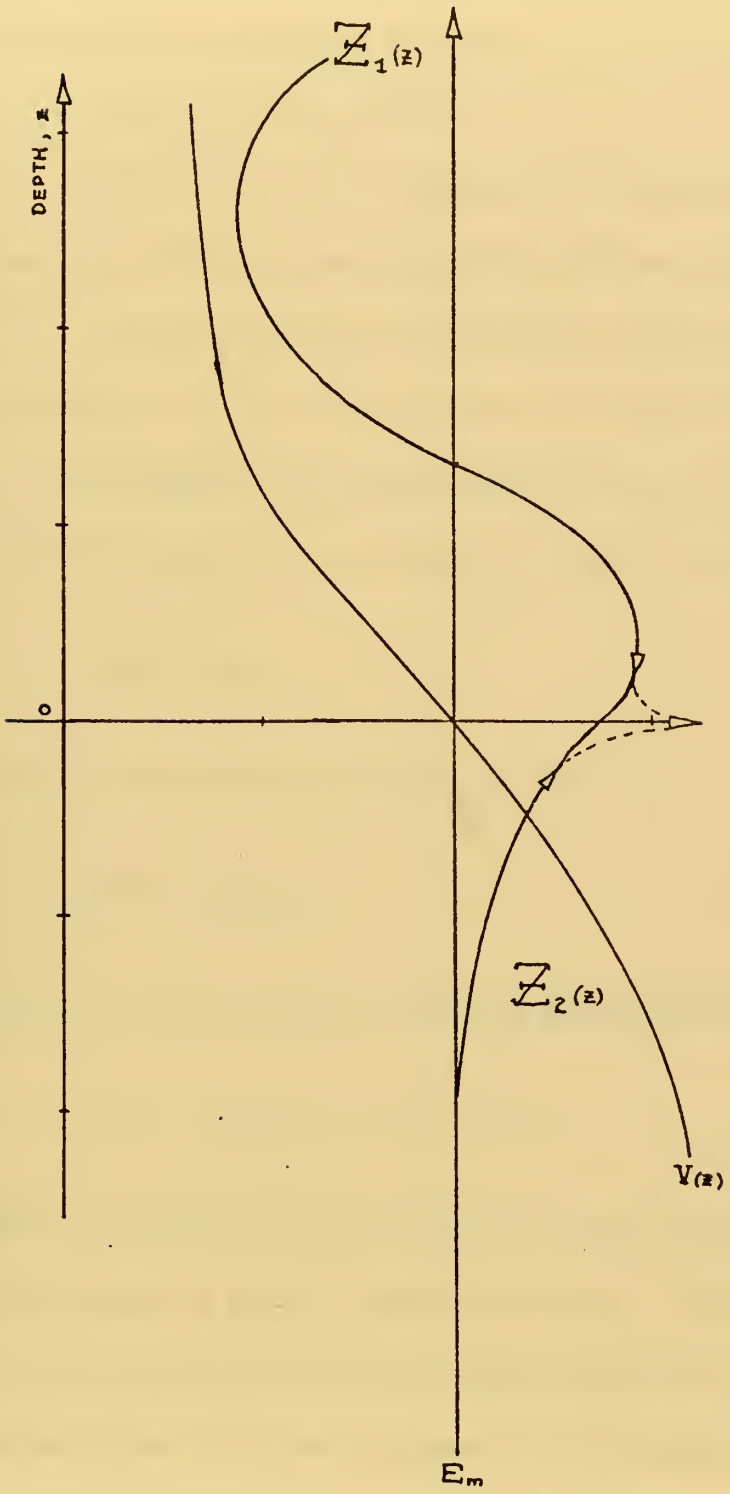


Figure 6

however, since it provides an interesting comparison with the Airy phase method in the computer programs.

b. The Airy Phase Method

What follows is an outline of a technique developed by V. R. Lauvstad (1971) of the SACLANT ASW Research Centre.

Lauvstad assumes asymptotic solutions of the same form as equations (57) and (59) in regions remote from the turning points. The gradient of the square of the vertical wave number across the turning point is assumed to be linear, such that

$$[k_z]^2 = \alpha z + \ldots. \tag{72}$$

This reduces the vertical Helmholtz equation to

$$Z'' + \alpha z Z = 0. \tag{73}$$

One solution to this equation is the Airy phase function

$$Z = A A_i(-S(z)) + B B_i(-S(z)), \tag{74}$$

where S(z), a depth integration function, is positive in the "real" region, and negative in the "imaginary" region. The asymptotic forms of the Airy phase are then compared with our corresponding assumed solutions. We can represent the asymptotic form of the Airy phase for negative real argument (the "real" region) as

$$Z_1 \cong \frac{1}{\sqrt{z\pi}}\left[(A-B)\sin S(z) + (A+B)\cos S(z)\right], \tag{75}$$

and for positive real argument as

$$Z_2 \cong \frac{1}{\sqrt{\pi}} \left[\frac{A}{2} e^{S(z)} + B e^{-S(z)} \right]. \tag{76}$$

By matching the above coefficients with those in our solutions we have a set of coefficient matching equations representing the required turning point connection.

$$a = \frac{1}{\sqrt{2}}(2D + C) \qquad b = \frac{1}{\sqrt{2}}(2D - C), \tag{77}$$

$$C = \frac{1}{\sqrt{2}}(a - b) \qquad D = \frac{1}{\sqrt{2}}(a + b). \tag{78}$$

III. THE PROGRAM METHODS

Here we will discuss the basic methods used by the three programs (NORMO1, NORMO4, and NORMO5) presented as a part of this thesis. This section is intended as only a brief introduction to the programs and their characteristics. A following section will give an outline of the program algorithms.

A. NORMO1 - A FINITE DIFFERENCE TECHNIQUE

NORMO1 is essentially an adaptation of two shallow water programs developed by Kanabis (1972) and Newman and Ingenito (1972). For a given value of the horizontal wave number, k_r, or E, the program begins with the bottom boundary condition (equation 22), and steps through a finite difference scheme to the surface. The finite difference scheme, which is derived in appendix A, is from Kanabis (1972). The finite difference scheme is a third-order Newton forward-difference scheme which iterates both the eigenfunction and its derivative. Using this difference scheme, a search is made for those values of k_r, or E, which most closely satisfy the surface boundary condition (equation 19). Such values are the sought eigenvalues.

A shortcoming of this technique is that under certain circumstances the eigenfunction at the surface, $Z_m(0)$, tends to grow exponentially rather than to decay towards zero. Some mention of this phenomenon is made by both Kanabis (1972) and Newman and Igenito

(1972). Kanabis recommends decreasing the incremental search limits for the eigenvalue (which imposes a consequential increase in computation time) to overcome the phenomenon. Newman and Igenito set the mode value equal to zero at depths above the last mode crossing or minimum. It happens that the circumstances for this exponential growth, or degenerate solution, occur frequently in deep ocean acoustic profiles. Thus, a more detailed treatment of the phenomenon is required.

To understand this degenerate solution let us consider a typical deep ocean sound profile and our general forms for the eigenfunction (equations 34 and 35). Consider a mode whose eigenvalue, E_m, intersects the velocity function, V(z), at some depth, z_{TU}, below the surface. This is a common (in fact, the usual) occurrence for the lower modes in the deep ocean, where a deep sound channel and thermocline usually exist. As noted by Schiff (1955) and Lauvstad (1971), if the value of S(z) in equation (34) is not exactly $\frac{\pi}{4}$ at the turning point, the coefficients C and D of equation (35) are both non-zero. Thus, the exponential growth term has some finite value. Given both sufficient depth between the turning point and the surface, plus sufficiently large values of [V(z)-E], the growth exponential will eventually become dominant. Thus, the solution tends to grow exponentially as the surface is approached.

This degenerate solution can complicate the search for eigenvalues if it is based solely upon the value of Z(0). With the

degenerate solution, the value of Z(0) will oscillate between extremely large positive and negative numbers, even with only small changes in the horizontal wave number or E. For the lowest modes of a deep sound channel profile the values of Z(0) can so oscillate with incremental changes in E of only one part in 10^{-14}. Thus, a method such as regula falsi (described later) can become inappropriate since a map of Z(0) as a function of E can appear discontinuous (Figure 7).

In order to circumvent this difficulty, NORMO1 uses a halving procedure based upon the number of mode crossings. When the surface boundary condition is met, the eigenvalue E_m is the largest possible value of E such that the eigenfunction has m-1 mode crossings. Based on this property, the number of mode crossings is used to converge upon the mode eigenvalue. The method of successive halving, although not sophisticated, has the versatility to deal succesfully with the degenerate solution and any arbitrary profile. Once an eigenvalue is found, the function Z_m is then checked for a degenerate solution near the surface. If it is present a new profile is iterated in the upper "imaginary" region and matched to the lower profile. This procedure may cause a discontinuity in the derivative of Z_m at the turning point; however, its ill effects would usually be less serious than those of the degenerate solution.

B. NORMO4 AND NORMO5 - THE WKB APPROACH

In both NORMO4 and NORMO5 the vertical wave number is integrated between the upper turning point (or surface) and lower turning

SURFACE MODE FUNCTION
Vs.
HORIZONTAL WAVE NUMBER

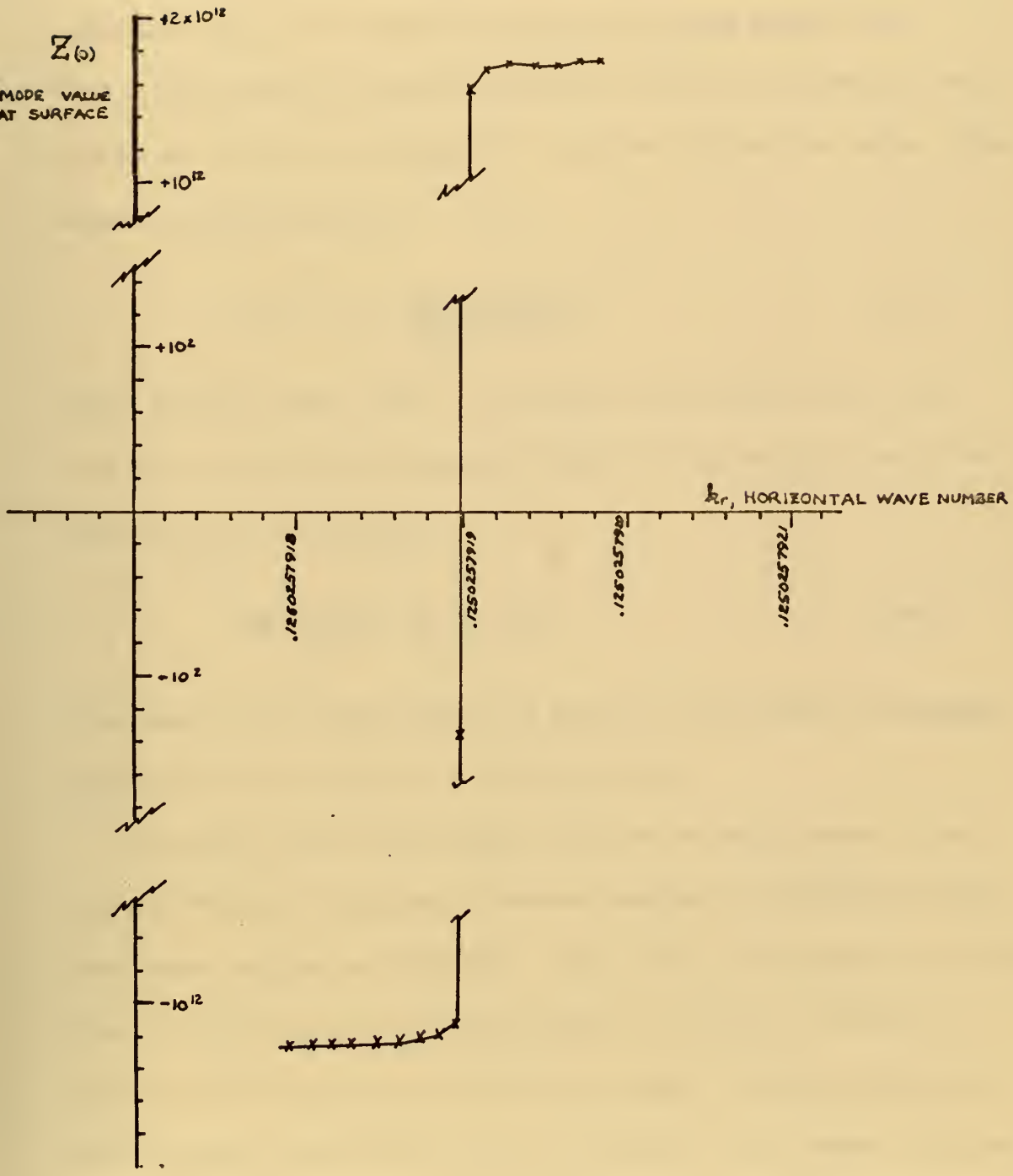


Figure 7

point (or bottom). The programs differ only in the turning point connection formulae. Both programs integrate the vertical wave number using the trapezoidal rule over the depth grid points. The eigenvalue E_m is converged upon using the regula falsi method. This method makes successive estimates of the zero value of a function by the linear interpolation of a negative and positive value. The method is represented by

$$E_{i+1} = E_- + \frac{(E_+ - E_-)(-\Delta S_-)}{(\Delta S_+ - \Delta S_-)} , \tag{79}$$

where the subscripts + and - correspond to the values associated with the last positive and negative values of ΔS; and ΔS, the difference function, is defined by

$$\Delta S = \int_{z_n}^{z_{n+1}} k_z \, dz + \theta_u + \theta_L - m\pi . \tag{80}$$

If the mode is not found after a set number of iterations, the regula falsi method is dropped for a halving process.

The phase effect of the upper and lower turning points, θ_u and θ_L, are evaluated using the asymptotic method for NORMO4 and the Airy phase method for NORMO5. The vertical wave number is integrated from the uppermost turning point (or surface, if surface reflected) to the lowest turning point (or bottom, if bottom reflected). Both programs must then provide, in addition to the values of θ_u and θ_L, the effect of any intermediate "imaginary" region or barrier. This occurs when multiple sound channels are encountered (Figure 8).

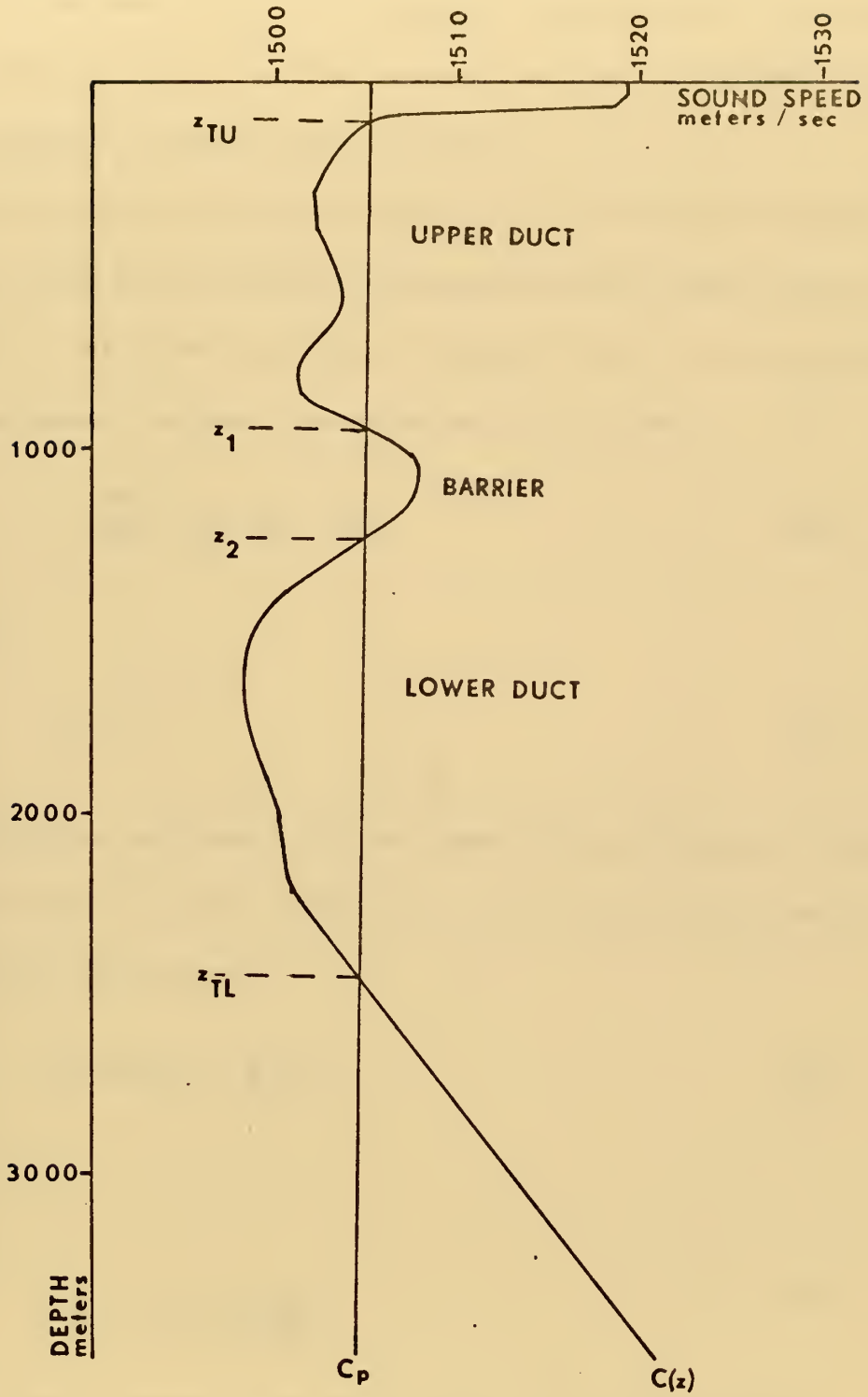

1500
1510
1520
1530
SOUND SPEED
meters / sec
z_{TU}
UPPER DUCT
z_1
BARRIER
z_2
LOWER DUCT
z_{TL}
1000
2000
3000
DEPTH
meters
C_p
C(z)

Figure 8.

Details of the derivation of the specific formulae for θ_u, θ_L, and the phase effect of an intermediate barrier are given in appendices B and C for NORMO4 and NORMO5 respectively.

If the mode is surface reflected or bottom reflected, both programs use the same formulae for the phase effect of the reflection, θ_u and θ_L. Since the free surface requires Z(0) = 0, for the surface reflected wave, we have from equations (63) and (66)

$$\int_{z_{TL}}^{z_{TU}} k_z \, dz + \theta_L = \theta_S = m\pi \tag{81}$$

and thus

$$\theta_u = 0 . \tag{82}$$

For the bottom reflected mode, we require the conditions of a fluid-fluid boundary to be met (equation 22). Substituting equations (63) and (25) into (22) we have

$$k_z \cot(\theta_L) = -\frac{\rho_0}{\rho_b} K_B , \tag{83}$$

or

$$\theta_L = \text{Arctan}\left(\frac{\rho_b}{\rho_0}\frac{k_z}{K_B}\right), \tag{84}$$

where k_z is evaluated at the water side of the bottom boundary.

The occurrence of multiple sound channels or ducts makes the WKB method more complicated. If the intermediate "imaginary" region or barrier is of sufficient extent and magnitude, sound

propagation in each channel becomes independent of the other. When this is the case, the waves in one channel undergo an effectively total reflection by the barrier. Thus, complete sound ducting occurs entirely within that channel. Lauvstad (1971) gives the ratio of the amplitudes of the reflected and incident waves in a duct as

$$R_1 = \frac{e^{L} - e^{-L}}{e^{L} + e^{-L}} , \tag{85}$$

where L is the integral of the imaginary vertical wave number across the barrier,

$$L = \int_{z_1}^{z_2} K_z \, dz . \tag{86}$$

This result can be derived using either the asymptotic or Airy phase connection formulae. As L increases, R_1 rapidly approaches unity, and near total reflection is achieved by the barrier. The WKB programs NORMO4 and NORMO5 evaluate L, and if it is of sufficient magnitude, the programs treat the mode propagation in each channel independently. The programs accommodate a maximum of one main sound channel (which contains the velocity minimum), and one upper and one lower sound channel.

IV. THE BASIC ALGORITHMS

A. NORMO1

1. General Description

NORMO1 iterates the wave equation vertically from the bottom to the surface using a finite difference scheme over a depth grid of up to 501 points. The program is written in IBM FORTRAN IV with subroutines used as major functional blocks. This modular programming is designed for both ease of modification and debugging. A description of the sequence of events follows.

NORMO1 is basically composed of three nested loops: a frequency loop, a mode loop, and an iteration loop. The input data are read using an input subroutine. The required input data are listed in appendix D. Once the data are read and appropriate constants computed, the sound velocity profile is linearly interpolated for all grid points. Here the frequency loop begins, the appropriate frequency is selected, and the velocity function V(z) is computed for the grid points. Next the maximum and minimum horizontal wave numbers and maximum number of modes are computed. If the frequency is below cut-off (the maximum number of modes is zero), a new frequency is selected. At this point the mode loop begins, and the following steps are repeated for the first to the highest requested or possible mode.

First, a method of successive halving is used to find two values of the horizontal wave number k_r which have m-1 and m mode crossings. The desired eigenvalue must lie between these two values. Then, again using a halving process, the program converges upon the eigenvalue k_r. The mode is considered found when one of two conditions (regulated by an input parameter, IEX) is satisfied;

$$|Z(0)| \leq |Z|_{max} 10^{-IEX}, \tag{87}$$

or the horizontal wave number changes between iterations by an amount such that

$$\left| \frac{k_{r_{i+1}} - k_{r_i}}{k_{r_i}} \right| < 10^{-2\,IEX}. \tag{88}$$

Once the mode is found, the eigenfunction is checked for a degenerate solution (exponential growth near the surface). If a degenerate solution is present, a new function is iterated from the surface to the upper turning point, then joined with the lower function. The mode is then normalized to its absolute maximum value, and the profile within the homogenous bottom is computed. The computed mode is plotted using a printer plot routine. A subroutine then computes the normalization integral represented by equation (41), and the group speed, phase speed and mode attenuation coefficient. At this point the mode and frequency loops end.

Included in NORMO1 is an output subroutine which provides a punched card format listing of the eigenvalues, mode parameters,

sound velocity profile and mode profile. The output subroutine was written to provide local input to CALCOMP plots and propagation loss programs.

2. The Subroutines

The following is a brief description of the major subroutines which comprise the NORMO1 program.

INPUT1 provides the input data for the program. A user supplied page heading is read on the first card, followed by the run parameters, list of frequencies desired, and a maximum of 50 depth and sound velocity pairs. If the parameter representing the number of depth and velocity pairs, NUMV, is negative, all length measurements read are converted from feet to meters. This subroutine also prints the input data for reference.

INTRPL linearly interpolates the input depth and sound velocity pairs to compute the grid point values of sound velocity. If required, this subroutine also extrapolates the last sound velocity to the bottom assuming an isothermal, isohaline gradient ($0.017 sec^{-1}$).

MAXMIN computes and checks the maximum and minimum values of the horizontal wave number, and finds the number of modes represented by the minimum horizontal wave number (the maximum number of modes). If the maximum number of modes is zero, the frequency is below cut-off and the program selects a new frequency or terminates the calculations. If the maximum possible number of modes is below that requested, this parameter is changed and noted.

HALF searches for two values of the horizontal wave number with m (the mode being searched) and m-1 mode crossings. These values represent lower and upper bounds on k_r which are then successively narrowed. At the same time this subroutine "maps" the values of k_r versus mode number in order to facilitate the search for later modes.

ITERAT, the heart of the program, iterates the eigenfunction from the bottom to the surface using equations (97) and (101). In addition, this subroutine counts the number of mode crossings and finds the maximum absolute value of the eigenfunction.

DNORM1 normalizes the eigenfunction with respect to its maximum absolute value.

BOTTOM creates an exponentially decaying bottom mode profile from the bottom to an input depth, PLTMAX.

MODPLT and CZPLOT are printer plot subroutines which plot the mode shape and sound velocity profile. Both these subroutines use a Naval Postgraduate School utility printer plot subroutine, UTPLOT.

FIXUP tests for a degenerate solution near the surface. If the degenerate solution exists, this subroutine applies the finite difference iteration scheme starting from the surface downward to the upper turning point. This upper mode shape is then scaled to join the original lower function at the turning point. If required, a new value for the absolute maximum of the eigenfunction is then found.

INTEGR computes the mode normalization integral (equation 41) and the integral represented by (equation 42). The phase velocity is computed; then the two integrals are used to compute the group velocity. Finally, the mode attenuation coefficient (equation 45) is calculated.

B. NORMO4 AND NORMO5

1. General Description

NORMO4 and NORMO5 are identical with the exception of the turning point connection formulae (appendices B and C explain these differences). Similar to NORMO1, they consist of three nested loops; a frequency loop, a mode loop, and an iterative loop represented by the subroutine SEARCH. In addition, the input, interpolation, plotting, maximum/minimum and mode integration functions are performed by subroutines similar to those used in NORMO1.

The sequence of events in the WKB method programs is similar to NORMO1. NORMO4 and NORMO5 read the input data and then interpolate the sound velocity profile over a maximum of 1001 depth grid points. If requested, a sound velocity printer plot or punched card format sound profile is provided. The frequency loop then computes the velocity function V(z) and checks for maximum horizontal wave number, minimum horizontal wave number, frequency cut-off, and the maximum number of modes available. At this point the mode loop begins and the search for the eigenvalue is controlled by the SEARCH subroutine.

After the mode is found, the mode shape is computed using a finite difference scheme. The finite difference scheme is fitted between the upper and lower turning points, then a solution which decays away from the turning points is calculated. After the mode shape has been calculated and normalized, the mode is integrated to provide the normalization integral, phase and group speed, and the mode attenuation coefficient. Finally, if requested, a printer plot of the mode shape is provided. At this point the mode and frequency loops end.

Because of their importance to these programs, the SEARCH and WKB subroutines deserve more extensive description.

2. SEARCH Subroutine

The SEARCH subroutine controls the iterative search for the eigenvalues k_r. The subroutine first makes an initial guess of an upper and lower bound for the eigenvalue. The WKB subroutine is then called to compute the difference function given by equation (80). If the difference function for the lower bound is not positive, the bound is successively relaxed until a positive value is found.

The next guess for the eigenvalue is calculated using a regula falsi zero-finding technique (given by equation 79). The resulting incremental change in eigenvalue and the absolute value of the difference function is checked to test whether the desired accuracy has been achieved. The WKB subroutine is then called to find the difference function for the new trial eigenvalue. If the resulting difference function is negative the upper bound is replaced; if it is

positive the lower bound is replaced. A new guess for k_r is then estimated, and this iterative process continues until one of two conditions is met. The mode is considered found when the difference function falls below an absolute limit,

$$|\Delta S| < 10^{-IEX} \tag{89}$$

or the value of the horizontal wave number approaches machine accuracy,

$$\left|\frac{k_{r_{i+1}} - k_{r_i}}{k_{r_i}}\right| \leq 10^{-14} \tag{90}$$

If after a set number of iterations, the regula falsi method has failed to converge upon a solution, it is replaced by a cruder and slower (but more versatile) halving process.

If more than one sound channel exists, the program must determine if the two channels are independent of one another. If the L of equation (86) is of sufficient magnitude, the subroutine SEARCH computes the eigenvalues for each channel separately. Thus, after an eigenvalue has been found, two tests are made. First, it is determined whether an upper or lower (separated) sound channel exists. Then, if an upper or lower channel exists, it is tested to determine whether it will propagate an additional mode. If such a mode can be propagated, the SEARCH subroutine computes its next mode within that channel.

3. WKB Subroutine

The purpose of the WKB subroutine is to integrate the vertical wave number between the upper and lower turning points. The subroutine consists of a loop which steps through the depth grid from the surface to the bottom. In any "imaginary" region the imaginary vertical wave number K_z is integrated for use with the turning point connection formulae. Each time the depth loop exits an "imaginary" region for a "real" region, either the upper turning point phase θ_u or the phase effect of a barrier must be calculated. If the barrier is considered to be separated (if L in equation 86 is larger than a set value) the integration is stopped with the upper channel.

When the depth loop reaches the bottom, the bottom phase, θ_L, is computed for either a bottom reflected or a decaying solution as appropriate. Finally, the difference function is calculated and program control returns to the calling program.

V. RESULTS

All three programs were run with a variety of sound velocity profiles and the calculated eigenvalues examined. The profiles selected include the published results of other models and profiles with known analytic solutions. In all the test runs the input variable IEX was set at 10. Thus, the iterations were halted when the absolute relative value of the eigenfunction at the surface, $Z_{(o)}$, or the difference function, ΔS , (equation 80) achieved a value less than 10^{-10}. Additionally, computation was halted if the eigenvalue was incremented less than 10^{-14} during a halving process (in other words, when the solution approached machine accuracy).

A. TEST CASE ONE - NEGATIVE GRADIENT

This test case is one given by Kanabis (1972). The profile is characterized by a simple negative velocity gradient in a shallow water channel (Figure 9). The profile has a fast fluid bottom of the same density as the water. The test results are computed for a frequency of 1000 Hz. Table 1 lists the eigenvalues given by Kanabis and computed by NORMO1, NORMO4 and NORMO5 in yd^{-1}. The list includes a set computed by the Kanabis program and a set computed by an unpublished program by C. L. Bartberger and L. L. Ackler. Except for the higher modes of NORMO5, the results from all three programs agree to the fourth decimal place or one part in 10^5.

TEST CASE ONE SOUND PROFILE

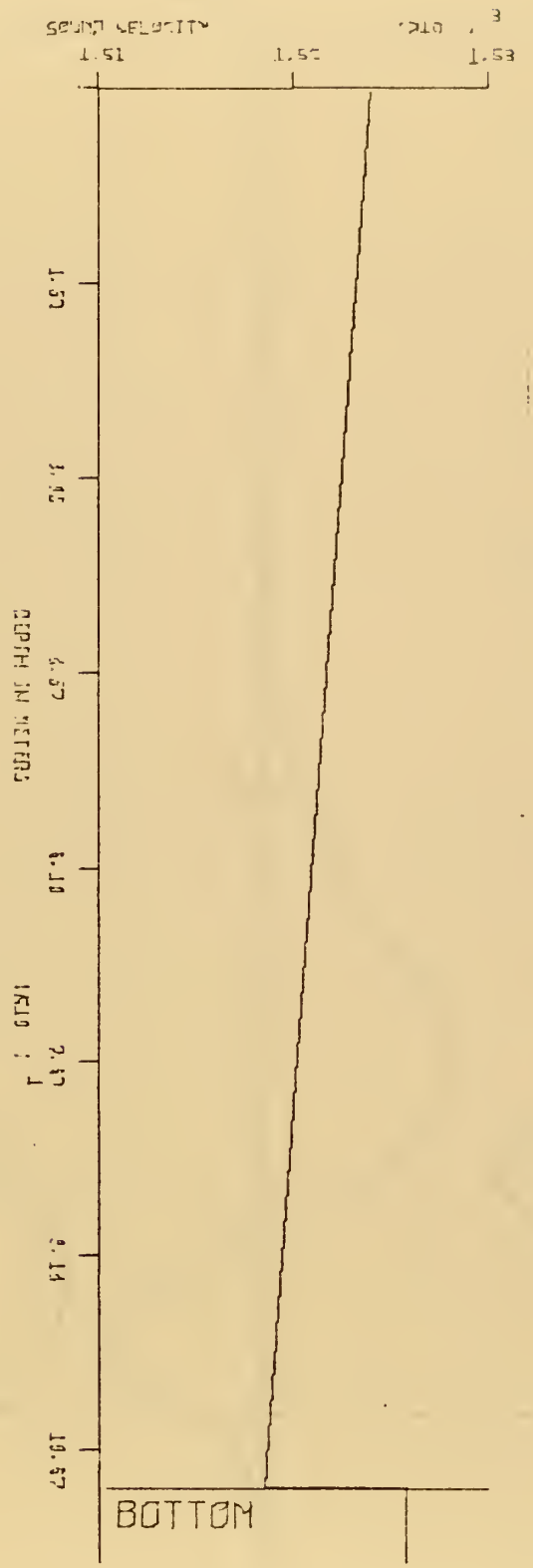


Figure 9

TEST CASE ONE MODES

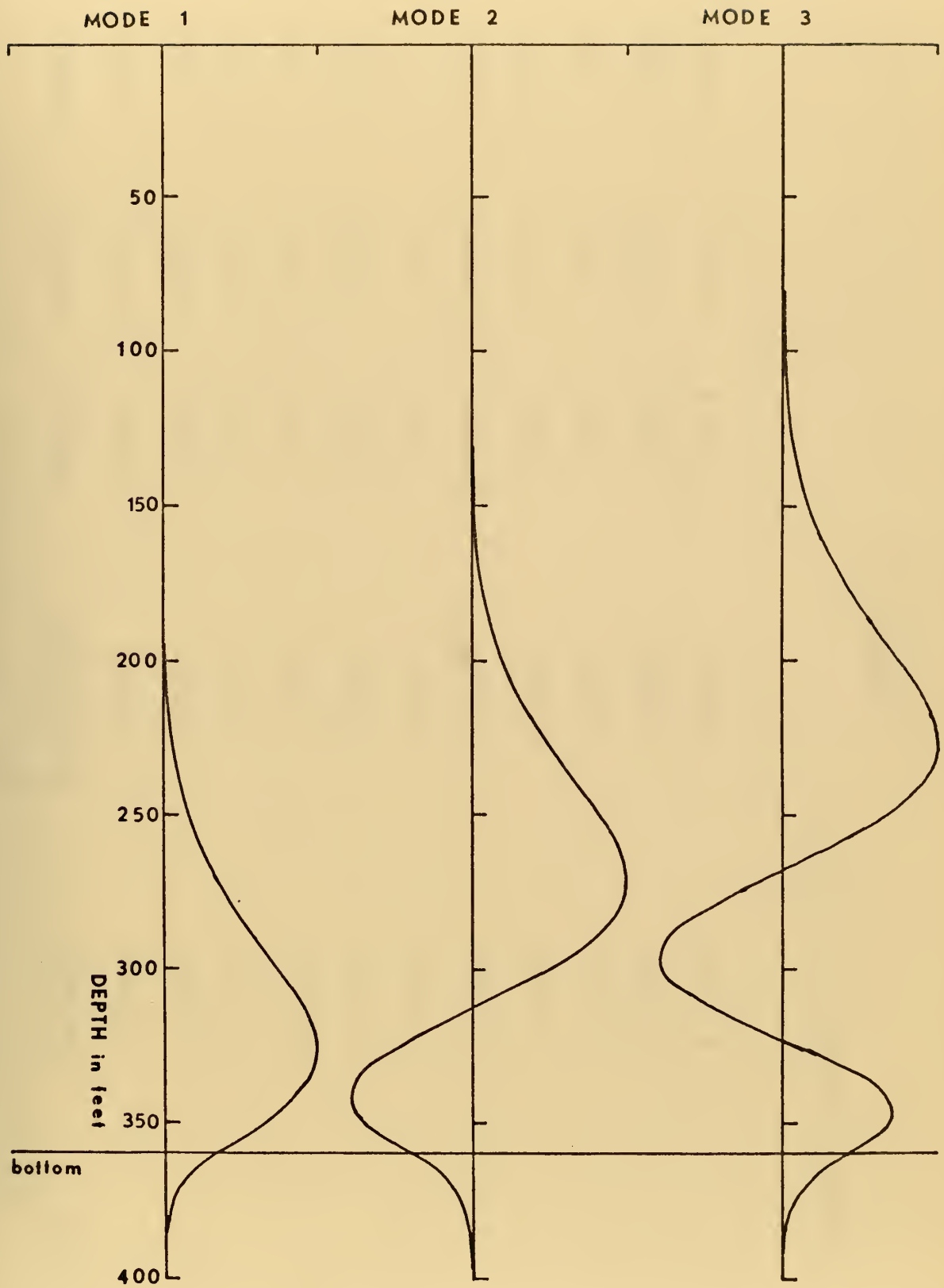


Figure 10

RESULTS OF TEST CASE ONE

MODE	KANABIS	BARTBERGER & ACKLER	NORMO1	NORMO4	NORMO5
1	3.78104	3.78105	3.78105	3.78107	3.78107
2	3.77896	3.77896	3.77896	3.77898	3.77898
3	3.77725	3.77725	3.77726	3.77727	3.77727
4	3.77574	3.77575	3.77575	3.77576	3.77576
5	3.77436	3.77437	3.77437	3.77438	3.77438
6	3.77309	3.77309	3.77309	3.77310	3.77310
7	3.77186	3.77187	3.77187	3.77187	3.77188
8	3.77063	3.77065	3.77065	3.77065	3.77065
9	3.76932	3.76933	3.76933	3.76930	3.76943
10	3.76782	3.76784	3.76784	3.76785	3.76801
11	3.76617	3.76618	3.76618	3.76621	3.76638

Horizontal wavenumbers in yard $^{-1}$.

TABLE I.

Table 2 gives the NORMO1, NORMO4 and NORMO5 results to the accuracy computed and relative differences between NORMO1 and the other two programs. NORMO1 required 56.23 sec of central processing unit (CPU) time and 412 iterations to arrive at the solutions. NORMO4 and NORMO5 required 23.50 seconds with 83 iterations and 22.74 seconds with 79 iterations respectively. The CPU times cited include the time required for all operations including input, sound speed profile manipulation and graph printing.

B. TEST CASE TWO - POSITIVE GRADIENT

The second test case is a shallow water example cited by Newman and Ingenito (1972). The profile is characterized by a slow, variable positive sound speed gradient over a fast, dense bottom (Figure 11). The computation frequency was 700 Hz. It should be noted that the accuracy parameter used by Newman and Ingenito was an absolute value of less than 0.001 for the surface eigenfunction Z(0). This roughly corresponds to a value of IEX between 3 and 4 in the NORMO programs. With the exception of NORMO5, all the programs agreed to at least the third decimal place (Table 3). Considering the accuracy limit set for the Newman and Ingenito run, this agreement is about as good as may be expected. NORMO1 required 27.43 seconds and 196 iterations to complete computations, while NORMO4 and NORMO5 required 14.73 seconds for 77 iterations and 14.92 seconds for 79 iterations, respectively.

NORMO SOLUTIONS RELATIVE DIFFERENCE

MODE	NORMO1	NORMO4	DIFF x 10^6	NORMO5	DIFF x 10^6
1	3.781048	3.781073	25	3.781073	25
2	3.778962	3.778977	15	3.778977	15
3	7255	7268	13	7268	13
4	5747	5760	13	5760	13
5	4370	4383	13	4383	13
6	3087	3100	13	3101	14
7	1869	1871	2	1882	13
8	0653	0558	105	0655	2
9	3.769331	3.769298	33	3.769432	101
10	7836	7846	10	8005	164
11	6179	6206	27	6380	201
CPU TIME:	56.23 sec	23.50 sec		22.74 sec	

TABLE II.

TEST CASE TWO PROFILE

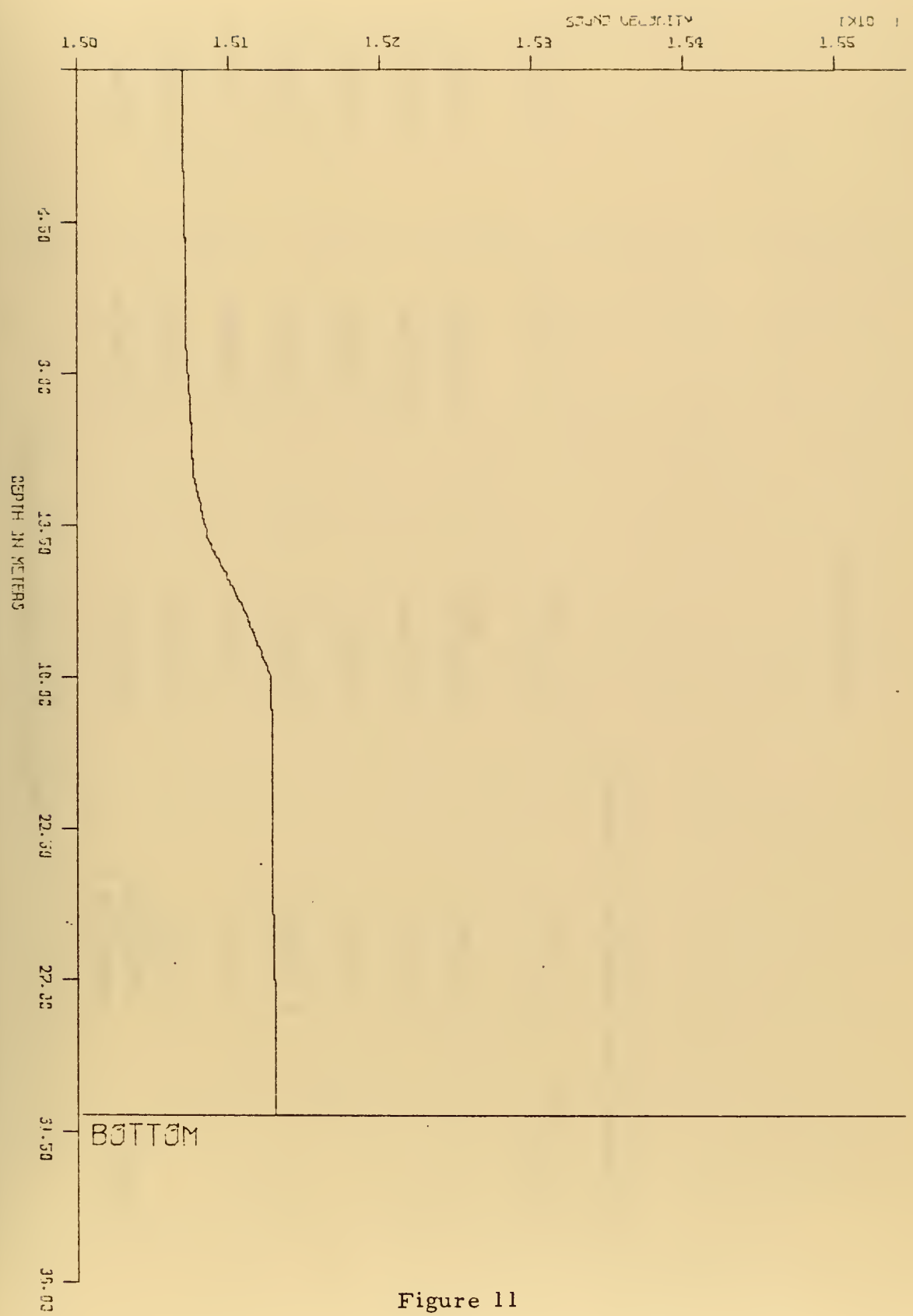


Figure 11

RESULTS OF TEST CASE TWO

MODE	NEWMAN & INGENITO	NORMO1	NORMO4	NORMO5
1	2.913214	2.913225	2.913215	2.914243
2	2.904392	2.904391	2.904848	2.905352
3	2.897892	2.897861	2.897634	2.898545
4	2.886656	2.886575	2.886816	2.888082
5	2.872573	2.872444	2.872565	2.874182
6	2.854842	2.854637	2.854866	2.856839
CPU Time:	27.43 sec	27.43 sec	14.73 sec	14.92 sec

Horizontal wave numbers in meters $^{-1}$

TABLE III.

C. TEST CASE THREE - SYMMETRIC WAVE DUCT

A profile was constructed corresponding to a symmetric wave duct where the sound speed is given by

$$\frac{1}{c^2} = \frac{1}{c_o^2} - \alpha^2\omega^2(z_o - z)^2 \qquad (115)$$

between two depths. The solution to this profile is given by Tolstoy and Clay (1966) as

$$k_{r_m} = \left[\frac{\omega^2}{c_o^2} - \omega\alpha(2m-1)\right]^{1/2} \qquad (116)$$

The actual profile dimensions, shown in figure 12, are taken from Tolstoy and Clay. The profile was tested at 30.0 and 60.0 hertz. The computed eigenvalues agree with the anlytic solutions to one part in 10^5, with the exception of the higher modes of NORMO1. Table 4 shows comparisons between the analytic solution and NORMO1, NORMO4 or NORMO5 computed values.

Of significant interest is the fact that the errors appeared relatively stable for NORMO4 and NORMO5. The errors for NORMO4 are consistently $0.45\text{x}10^{-6}$ to $0.69\text{x}10^{-6}$ meters $^{-1}$ low for 30 Hz, and $0.10\text{x}10^{-5}$ to $0.12\text{x}10^{-5}$ meters $^{-1}$ low for 60 Hz. In contrast, the errors for NORMO1 increased from $0.39\text{x}10^{-6}$ meters $^{-1}$ low to $12.5\text{x}10^{-6}$ meters $^{-1}$ high for 30 Hz, and from $0.80\text{x}10^{-6}$ to $2.8\text{x}10^{-6}$ meters $^{-1}$ low for 60 Hz. Figure 14 displays a graph of the program errors. Figure 13 shows the first three modes.

TEST CASE THREE PROFILE

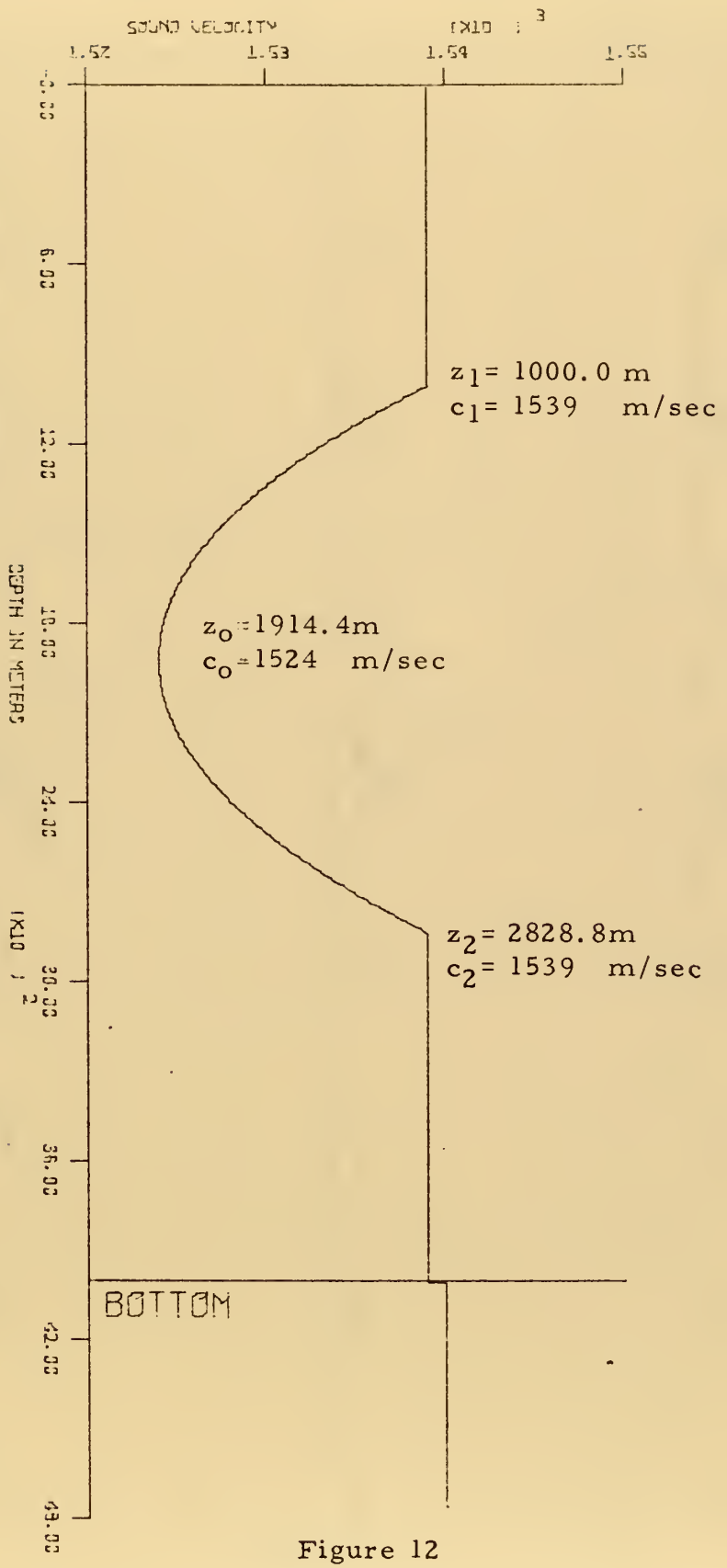


Figure 12

TEST CASE THREE MODES

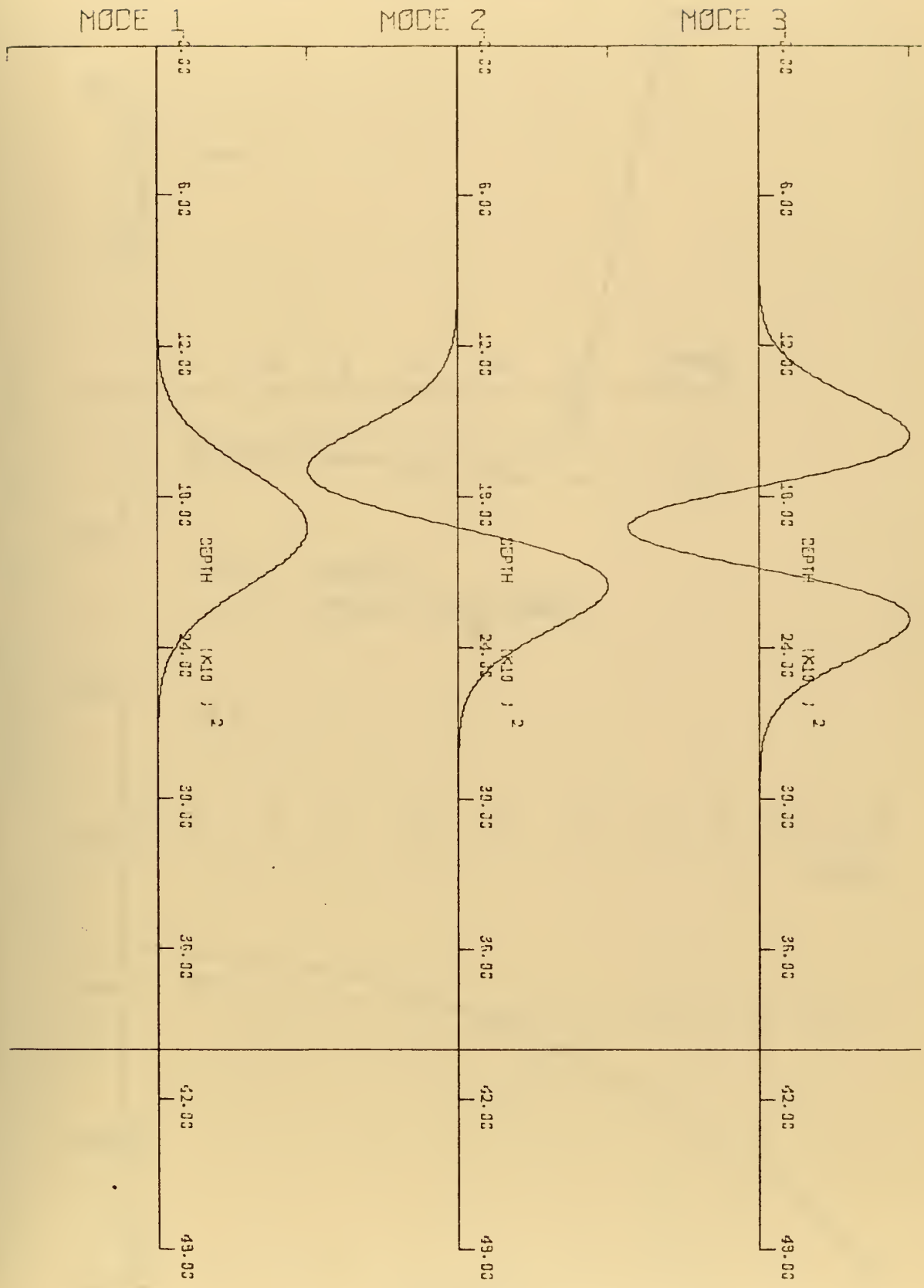


Figure 13

TEST CASE THREE ERRORS

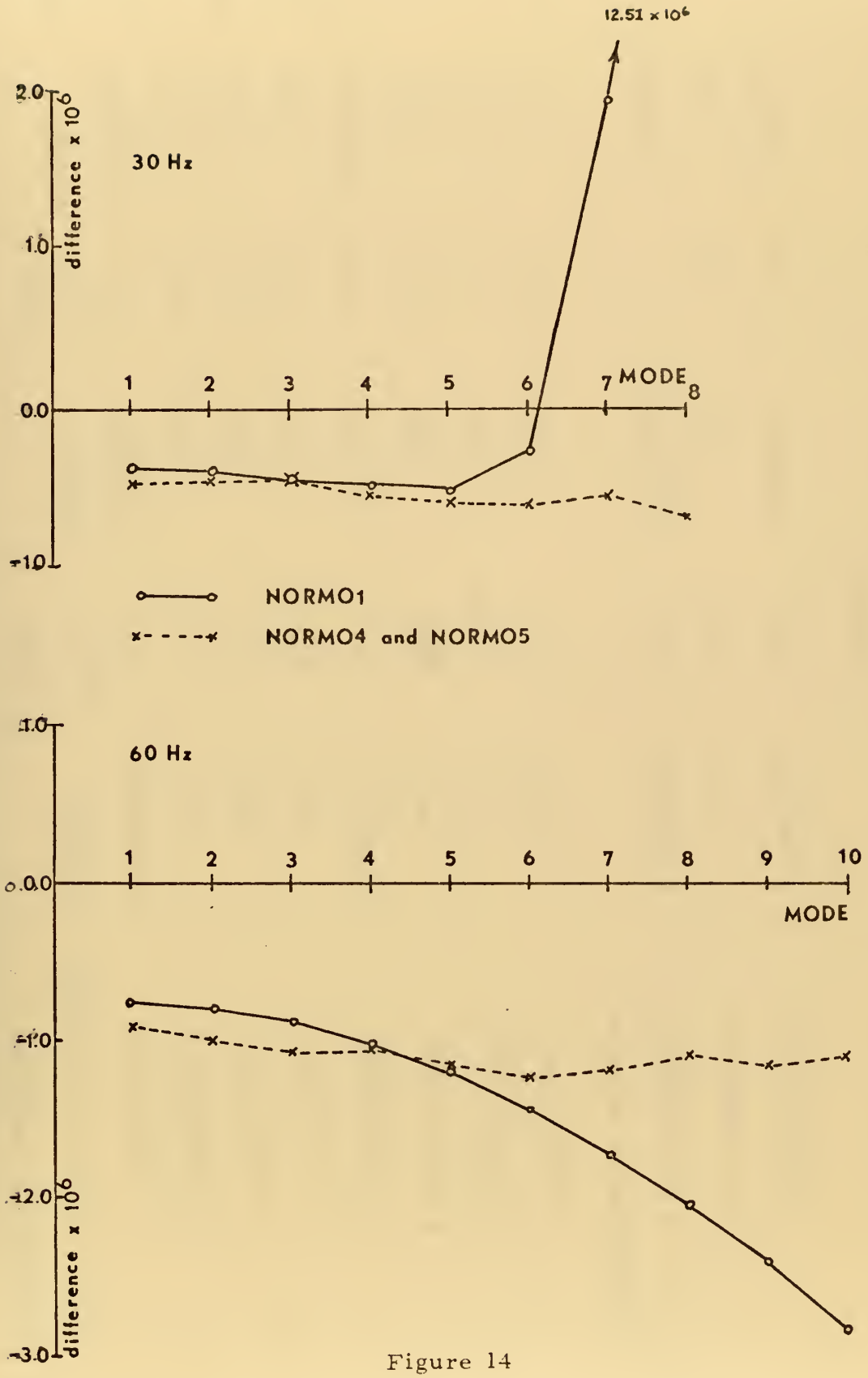


Figure 14

RESULTS OF TEST CASE THREE

MODE	ANALYTIC SOLUTION	NORMO1	DIFF $\times 10^{-6}$	NORMO4	DIFF $\times 10^{-6}$	NORMO5	DIFF
FREQUENCY: 30 hertz							
1	0.123608568874	0.1236081	-0.37	0.1236080	-0.49	0.1236080	-0.49
2	0.123456065357	4556	-0.39	4555	-0.48	4555	-0.48
3	0.123303373221	3029	-0.44	3028	-0.45	3028	-0.44
4	0.123150491766	1499	-0.50	1498	-0.55	1498	-0.50
5	0.122997420284	0.1229969	-0.52	0.1229968	-0.60	0.1229968	-0.52
6	0.122844158066	8439	-0.27	8435	-0.61	8435	-0.27
7	0.122690704396	6926	+1.966	6901	-0.56	6901	-0.56
8	0.122537058556	5495	12.51	5363	-0.69	5363	-0.68
FREQUENCY: 60 hertz							
1	0.247293330726	0.2472925	-0.75	0.2472924	-0.91	0.2472924	-0.91
2	0.247140921278	1401	-0.80	1399	-1.00	1399	-1.00
3	0.246988417783	0.2469875	-0.88	0.2469873	-1.08	0.2469873	-1.08

TABLE 4

RESULTS OF TEST CASE THREE (Continued)

FREQUENCY: 60 hertz

MODE	ANALYTIC SOLUTION	NORMO1	DIFF x10-6	NORMO4	DIFF x10-6	NORMO5	DIFF
4	0.246835820066	0.2468348	-1.01	0.2468347	-1.03	0.2468347	-1.03
5	0.246683127953	6819	-1.20	6819	-1.15	6819	-1.15
6	0.246530341267	5289	-1.43	5291	-1.21	5291	-1.21
7	0.246377459834	3757	-1.71	3762	-1.18	3762	-1.18
8	0.246224483476	2224	-2.03	2234	-1.08	2234	-1.08
9	0.246071412016	0690	-2.40	0702	-1.15	0702	-1.15
10	0.245918245277	0.2459154	-2.82	0.2459171	-1.09	0.2459171	-1.09

TABLE 4

D. TEST CASE FOUR - DEEP SOUND CHANNEL

Test case four is a typical deep ocean sound channel. The profile is characterized by a deep and sharp sound channel and by a bottom of the same density as the water (Figure 15). With the exception of the first mode, the eigenvalues agree within one part in 10^4 (Table 5). Note that for the higher frequency, 60 Hz, the agreement is somewhat improved. It is interesting that the first mode should show a larger disagreement among the programs. This effect may be a result of the combination of the sharp sound channel axis and a grid spacing of 8 meters.

Figure 15

RESULTS OF TEST CASE FOUR

MODE	NORMO1	NORMO4	DIFF x10^6	NORMO5	DIFF x10^6
Frequency: 30 hertz					
1	0.12614128	0.12610844	-32.84	0.12610844	-32.84
2	567736	567933	- 1.97	567933	- 1.79
3	537387	538187	8.00	538187	8.00
4	514437	513923	- 5.14	513923	- 5.14
5	492671	492224	- 4.47	492224	- 4.47
6	472443	472294	- 1.49	472294	- 1.49
7	453745	453659	- 0.86	453660	- 0.85
8	436162	436012	- 1.50	436018	- 1.44
9	419601	419117	- 4.84	419155	- 4.46
10	404106	400638	-34.68	402169	-19.37
Frequency: 60 hertz					
1	0.25255354	0.25250819	-45.35	0.25250819	-45.35
2	196659	196827	1.68	196827	1.68
3	154658	154783	1.25	154783	1.25
4	118755	118986	2.31	118986	2.31
5	089177	089755	5.78	089755	5.78
6	063654	063571	- 0.83	063571	- 0.83
7	039561	039343	- 2.18	039343	- 2.18
8	016594	016618	0.24	016618	0.24
9	0.24994761	0.24994890	1.29	0.24994890	1.29
10	973743	974255	5.12	974255	5.12

TABLE V

E. TEST CASE FIVE - DOUBLE CHANNEL

Test case five is a double sound channel of nearly symmetric dimensions (Figure 16). This profile was used to test the multi-duct properties of the NORMO4 and NORMO5 programs. Note that for 30 Hz the differences between NORMO1 and NORMO4 (or NORMO5) increase by almost an order of magnitude for modes 5 and 7 (Table 6). For these modes the barrier connection formula (equation 114) is employed and the ducts are considered "connected." The mode profiles (Figure 17) for these modes also do not completely correspond. Modes 5 and 7 appear to be "upside-down" in NORMO4, with the dominant amplitude waves in the lower channel. This is due to the fact that NORMO4 and NORMO5 iterate the vertical wave number from the surface downward. Because of this, the phase at the upper turning point of the lower channel represents the phase angle for the third and fourth mode in the lower channel. Equation (114) remains correct, although here it has been incorrectly applied.

TEST CASE FIVE PROFILE

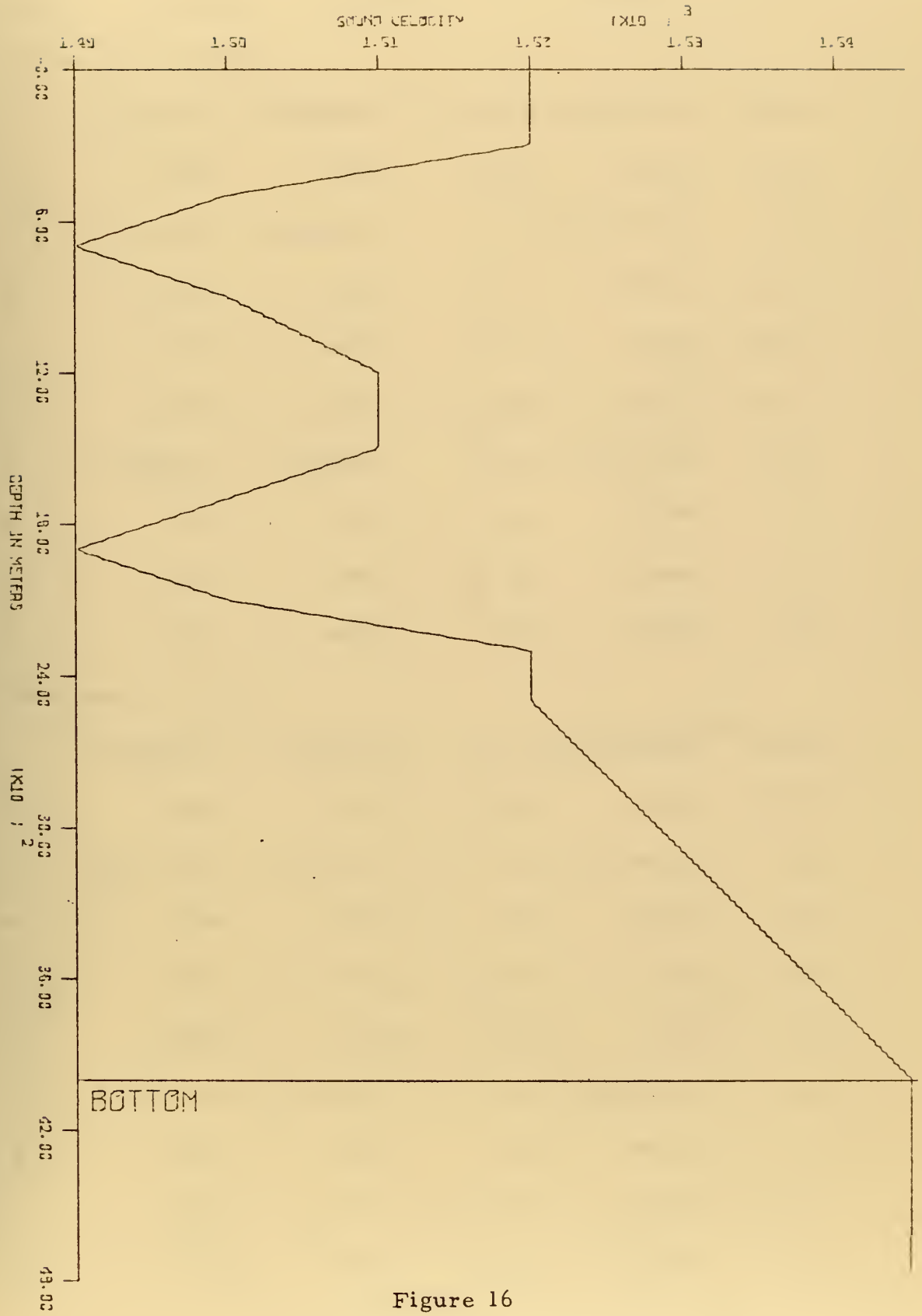


Figure 16

RESULTS OF TEST CASE FIVE

MODE	NORMO1	NORMO4	DIFF $\times 10^6$	NORMO5	DIFF $\times 10^6$
Frequency: 30 hertz					
1	0.12608464	0.12604450	-40.14	0.12604450	-40.14
2	08403	04450	-39.53	04450	-39.53
3	0.12553486	0.12554660	11.76	0.12554660	11.76
4	52366	53937	15.71	53937	15.71
5	16376	36343	199.67	36343	199.67
6	11170	09716	-14.54	09716	-14.54
7	0.12485241	0.12490245	50.04	0.12490245	50.04
8	75394	74793	- 6.01	74793	- 6.01
9	59913	59806	- 1.07	59806	- 1.07
10	42282	42565	2.83	42568	2.86
Frequency: 60 hertz					
1	0.25248206	0.25242998	-52.08	0.25242998	-52/08
2	48204	42998	-52.06	42998	-52.06
3	0.25179326	0.25180252	9.26	0.25180252	9.26
4	79260	80252	9.92	80252	9.92
5	31541	31134	- 4.07	31134	- 4.07
6	30922	31100	1.78	31100	1.78
7	0.25087788	0.25088532	7.44	0.25088532	7.44
8	84379	84790	4.11	84790	4.11
9	50385	49918	- 4.67	49918	- 4.67
10	41349	40770	- 5.79	40770	- 5.79

TABLE VI

DOUBLE CHANNEL MODE COMPARISONS

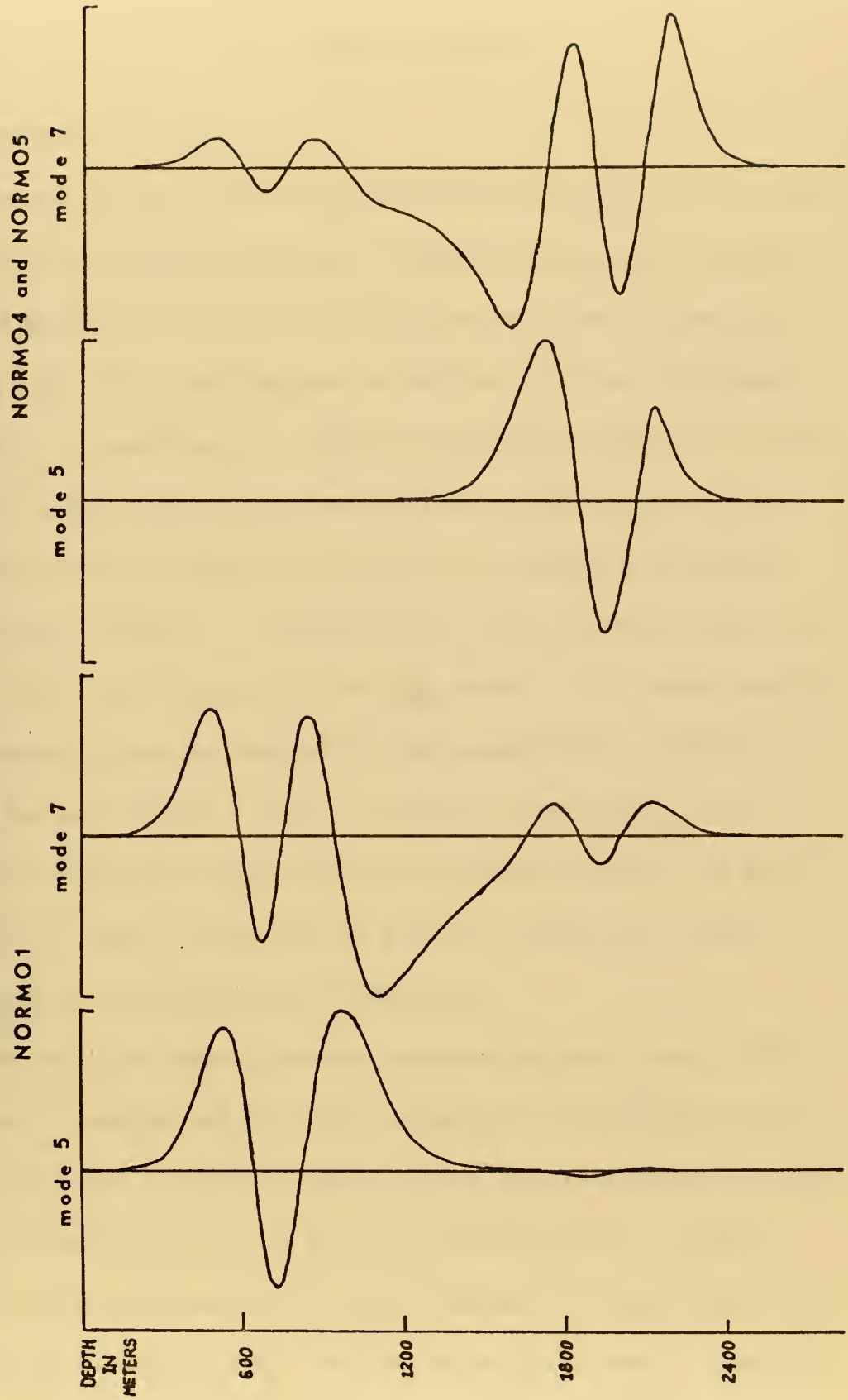


Figure 17

VI. CONCLUSIONS

A. GENERAL

As seen by table 7, the WKB method has performed consistently faster than the iterative technique. The WKB programs (NORMO4 and NORMO5) found solutions with about one-half the Central Processing Unit (CPU) time required by the finite difference program (NORMO1). In addition, one should consider the method used for integration of the vertical wave number in the WKB programs. The trapezoid rule was used which required the evaluation of a square root at each grid point. Although simple, this is a time consuming task, while it is not necessarily very accurate. If a faster integrating procedure were substituted for the trapezoid rule, the time required for each iteration could be reduced considerably. Thus, in addition to showing an improvement in computation time, the WKB method also offers the potential of greater computational speed through the use of sophisticated integrators.

The WKB and finite difference methods appear to have similar accuracy in comparison with other programs. The accuracy attainable is adequate to calculate useful modes for a propagation loss program. However, the errors involved in the differences between successive eigenvalues will not permit accurate coherent phasing of the modes beyond the first or second convergence zone . Coherent phasing at extremely long ranges is, in fact, attainable only with an

CPU TIMES FOR VARIOUS RUNS

	NORMO1	NORMO4	NORMO5
Test Case One	56.23 sec	23.50 sec	22.74 sec
Test Case Two	27.43	14.73	14.92
Test Case Three	106.35	38.42	39.47
Test Case Four	105.67	40.23	40.43
Test Case Five	114.07	44.53	47.09
TOTAL	409.75 sec	161.41 sec	164.65 sec

TABLE VII

analytic profile and solution — an ideal mathematical model. In fact, when one considers the accuracy of the available sound velocity data, it is difficult to imagine any model attempting to yield accurate phasing effects at any large distance. Since the position of a target is unknown, in practice it is more important that the amplitude and nature of phasing effects be know, than for the sharp interference peaks to be accurately predicted. Thus, accuracies of one part in 10^5 should enable a model both to take account of the phasing effects in the first convergence zone, and to give a general description of the effects of phase interference at longer ranges.

Although the WKB programs do not correctly interpret the barrier effects, the problem appears to lie in the approach taken in the programming, rather than in the mathematical and physical relationships. Within any duct, the effects of any upper and lower ducts must be treated as a turning point phase effect similar to the top and bottom boundary conditions. The modes 5 and 7 of the NORMO4 30 Hz double channel case are unwitting examples of this. In NORMO4 and NORMO5 the "connected" ducts are considered as a single wave system. In fact, with any sized barrier, no matter how small, one must consider the wave system within each duct separately. In this case the turning point phase angle between the duct and barrier is analogous to the impedence due to the other duct felt by the wave system in its duct.

B. FUTURE PROGRAM

It appears that a faster WKB method normal mode program can be written which will properly interpret a barrier. In order to properly consider the multiple channel cases, a mapping procedure should be used before actual mode searches commence. The sound profile would be divided into "levels" (Figure 18), found by selecting all the velocity maxima and minima. Each "level" represents a range of the possible eigenvalues, E_m. Within each "level" there is a single combination of ducts, barriers and boundaries. At the start of computations the program would determine the limits of each level. Then, at the beginning of the frequency loop, the program would determine the maximum and minimum number of modes for each duct within each level. Thus, equipped with a map of the profile characteristics, the program would compute eigenvalues for each duct and level in sequence (Figure 19). After all the eigenvalues are computed and sorted (some may be out of numerical order), the eigenfunctions can be computed quickly using the finite difference procedure of NORMO1. Combined with a faster integrating method, such a program should yield fast, accurate normal modes for an arbitrary sound speed profile.

SOUND VELOCITY PROFILE
WITH
LEVELS

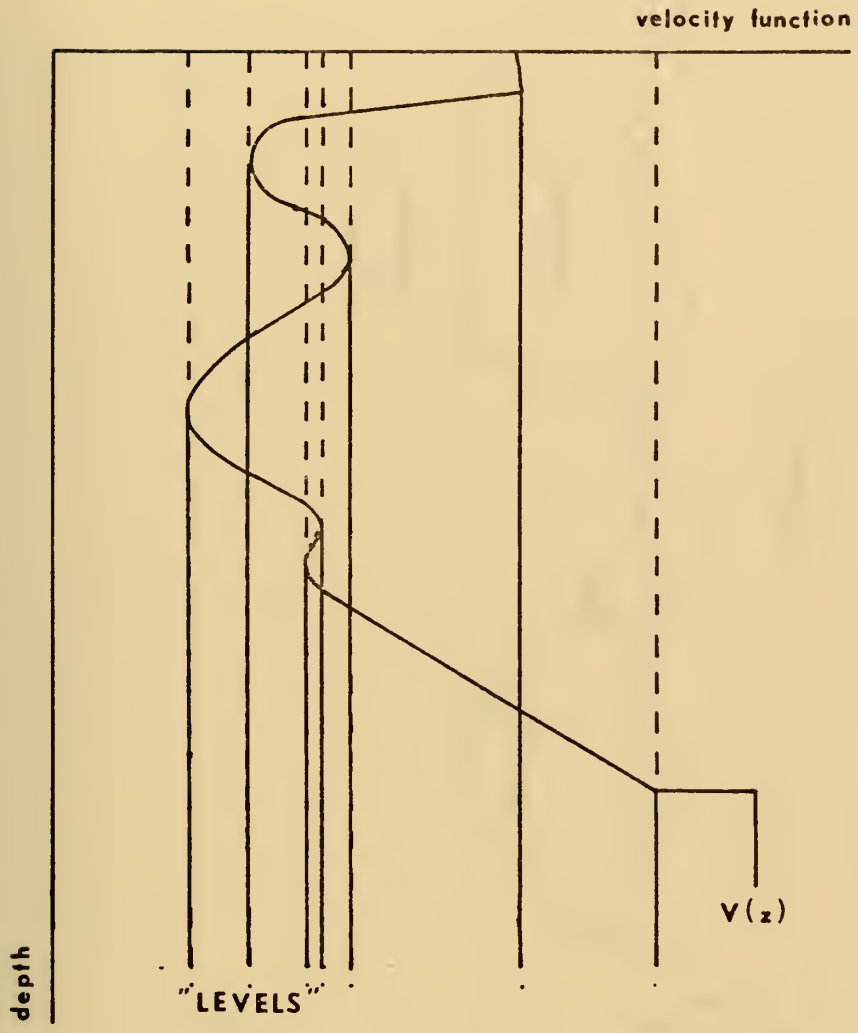
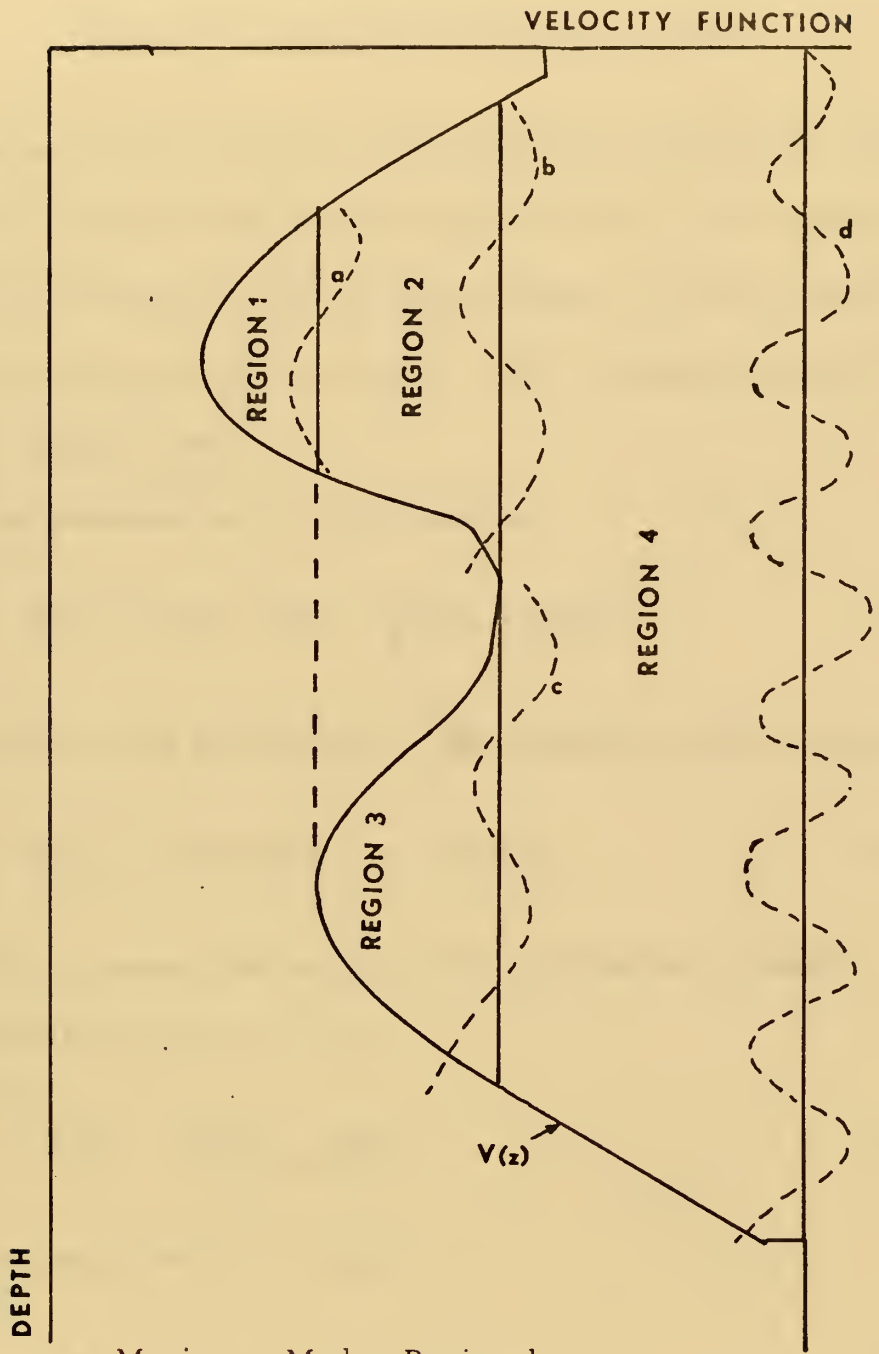


Figure 18

SEQUENCE OF MODE CALCULATIONS



a Maximum Modes Region 1

b Maximum Modes Region 2

c Maximum Modes Region 3

d Maximum Modes Region 4

Figure 19

APPENDIX A

NORMO1 FINITE DIFFERENCE SCHEME

In order to iterate equation (18) through the depth grid, a finite difference scheme is used which computes values of the eigenfunction, Z(z) and its derivative, Z'(z) at each grid point. This scheme was originally developed by the Arthur D. Little Company [Report No. C-70673, 31 January 1969].

Z(z) is expanded into a Taylor series,

$$Z_{(z+h)} = Z_{(z)} + hZ'_{(z)} + \frac{h^2}{2!}Z''_{(z)} + \frac{h^3}{3!}Z'''_{(z)} , \qquad (91)$$

where h is the depth grid spacing. From equation (18) we have

$$Z''_{(z)} = -\left[E - V_{(z)}\right] Z_{(z)} = -F_{(z)}Z_{(z)} , \qquad (92)$$

where F(z) represents the square of the vertical wave number. Using a forward difference for Z'''(z),

$$Z'''_{(z)} = \frac{Z''_{(z+h)} - Z''_{(z)}}{h} . \qquad (93)$$

Substituting equation (92), we have

$$Z'''_{(z)} = \frac{1}{h}\left[F_{(z)}Z_{(z)} - F_{(z+h)}Z_{(z+h)}\right] . \qquad (94)$$

Combining equations (91), (92) and (94),

$$Z_{(z+h)} = Z_{(z)} + hZ'_{(z)} - \frac{h^2}{2!}F_{(z)}Z_{(z)} + \frac{h^3}{3!}\left[\frac{F_{(z)}Z_{(z)} - F_{(z+h)}Z_{(z+h)}}{h}\right] , \qquad (95)$$

we have on clearing,

$$Z(z+h)\left[1+\frac{h^2}{3!}F(z+h)\right] = Z(z)\left[1-\frac{h^2}{3}F(z)\right] + hZ' . \qquad (96)$$

Placing the above result in a vector subscript notation we have the formula used to compute the succeeding eigenfunction,

$$Z_{i+1} = \frac{Z_i\left(1-\frac{h^2}{3}F_i\right) + hZ'_i}{\left(1+\frac{h^2}{6}F_{i+1}\right)} . \qquad (97)$$

We must now compute the value of the derivative of the eigenfunction Z'(z) at the next grid point. In order to do this let us express the eigenfunction at our original grid point as a backwards Taylor series of the new grid point, z+h,

$$Z(z) = Z(z+h) - hZ'(z+h) + \frac{h^2}{2!}Z''(z+h) - \frac{h^3}{3!}Z'''(z+h) . \qquad (98)$$

Substituting equations (92) and (93), we have

$$Z(z) = Z(z+h) - hZ'(z+h) - \frac{h^2}{2}F(z+h)Z(z+h) - \frac{h^2}{6}\left[F(z)Z(z) - F(z+h)Z(z+h)\right] . \qquad (99)$$

Rearranging, we have,

$$Z'(z+h) = \frac{1}{h}\left[Z(z+h)\left(1-\frac{h^2}{3}F(z+h)\right) - Z(z)\left(1+\frac{h^2}{6}F(z)\right)\right] , \qquad (100)$$

or in vector notation,

$$Z'_{i+1} = \frac{1}{h}\left[Z_{i+1}\left(1-\frac{h^2}{3}F_{i+1}\right) - Z_i\left(1+\frac{h^2}{6}F_i\right)\right] . \qquad (101)$$

APPENDIX B

NORMO4 CONNECTION FORMULAE

NORMO4 requires the evaluation of the phase at the upper and lower turning points in addition to the phase effect of a barrier. For the purposes of this discussion we will align the origin of the depth axis with the turning point, designating the positive direction towards the "real" region.

Consider the upper turning point case (Figure 5). The free surface boundary condition requires that the "imaginary" region solution disappear at the surface. From equation (59) this requires that

$$Z(-H) = K_z(-H)\left[Ce^{S_z(-H)} + De^{-S_z(-H)}\right] = 0. \tag{102}$$

Thus, we can define C and D (except for a constant of multiplication);

$$C = -e^{-S_z(-H)} \qquad D = e^{S_z(-H)}. \tag{103}$$

Applying equation (71), we have

$$\frac{a}{b} = \frac{e^{-S_z(-H)} - e^{S_z(-H)}}{e^{-S_z(-H)} + e^{S_z(-H)}}, \tag{104}$$

or

$$\theta_u = \text{Arctan}\left[\text{Tanh}\left(\int_{-H}^{0} K_z\, dz\right)\right]. \tag{105}$$

Now, consider the bottom boundary condition and lower turning point (Figure 5). The bottom boundary condition (equation 26) requires

$$\frac{\rho_o}{\rho_b} K_B = K_S \frac{C e^{S_z(-H)} - D e^{-S_z(-H)}}{C e^{S_z(-H)} + D e^{-S_z(-H)}} , \tag{106}$$

where K_s is defined as the imaginary vertical wave number evaluated at the water side of the bottom interface,

$$K_S^2 = [V(-H) - E] = k_r^2 - \frac{\omega^2}{c(-H)^2} . \tag{107}$$

Thus, the imaginary solution can be written by defining C and D such that

$$\begin{aligned} C &= (\rho_b K_S + \rho_o K_B) e^{-S_z(-H)} \\ D &= (\rho_b K_S - \rho_o K_B) e^{S_z(-H)} . \end{aligned} \tag{108}$$

Now, by applying the turning point connection formulae (equation 71) we have

$$\frac{a}{b} = \frac{(\rho_b K_S + \rho_o K_B) e^{-S_z(-H)} + (\rho_b K_S - \rho_o K_B) e^{S_z(-H)}}{(\rho_b K_S - \rho_o K_B) e^{S_z(-H)} - (\rho_b K_S + \rho_o K_B) e^{-S_z(-H)}} , \tag{109}$$

or

$$\theta_L = \text{Arctan} \left[\frac{D_z + 1}{D_z - 1}\right] , \tag{110}$$

where D_2 is defined as

$$D_2 = \frac{\rho_b K_s - \rho_o K_s}{\rho_b K_s + \rho_o K_s} e^{2S_2(-H)} \tag{111}$$

Now, consider the situation with an upper and lower channel separated by a barrier (Figure 8). Let the z origin be at the lower connection point, z_2, with the positive direction downward. From equations (59), (69) and (71) we have for a connection at the upper point, z_1,

$$\frac{a_1}{b_1} = \frac{De^{-L} + Ce^{L}}{De^{-L} - Ce^{L}} , \tag{112}$$

where L is as defined in equation (86). Letting A_1 equal the numerator and B_1 the denominator, we have, except for a constant,

$$\begin{aligned} C &= \tfrac{1}{2}(a_1 - b_1)e^{-L} \\ D &= \tfrac{1}{2}(a_1 + b_1)e^{L} . \end{aligned} \tag{113}$$

Applying these expressions to equation (71), we have

$$\frac{a_2}{b_2} = \frac{(a_1 + b_1)e^{L} + (a_1 - b_1)e^{-L}}{(a_1 + b_1)e^{L} - (a_1 - b_1)e^{-L}} , \tag{114}$$

or

$$\theta_2 = \text{Arctan}\left[\frac{(\tan\theta_1 + 1)e^{L} + (\tan\theta_1 - 1)e^{-L}}{(\tan\theta_1 + 1)e^{L} - (\tan\theta_1 - 1)e^{-L}}\right] . \tag{115}$$

APPENDIX C

NORMO5 CONNECTION FORMULAE

For NORMO5 the development of the upper and lower "imaginary" region particular solutions by the application of the free surface and bottom boundary conditions is identical to that for NORMO4. The differences in the two programs lie in the application of the Airy phase connection formulae vice the asymptotic connection formulae. To obtain the upper turning point phase, we apply equation (103) to equation (77)

$$a = \frac{1}{\sqrt{2}}\left(2\, e^{S_2(-H)} - e^{-S_2(-H)}\right) \tag{116}$$

$$b = \frac{1}{\sqrt{2}}\left(2\, e^{S_2(-H)} + e^{-S_2(-H)}\right), \tag{117}$$

or

$$\theta_u = \text{Arctan}\left[\frac{D_1 - 1}{D_1 + 1}\right], \tag{118}$$

where D_1 is defined as

$$D_1 = 2\, e^{2S_2(-H)} . \tag{119}$$

Note that as H approaches zero (the turning point approaches the surface) the value of θ_u in equation (119) approaches arctan (1/3) vice the phase we required for the free surface boundary condition, ($\theta_u = 0$).

To derive the lower turning point phase, we apply the particular lower "imaginary" region solution (equation 108) to the Airy phase connection formulae (equation 77),

$$a = \frac{1}{\sqrt{2}}\left[2(\rho_b K_S - \rho_o K_B)e^{S_2(-H)} + (\rho_b K_S + \rho_o K_B)e^{-S_2(-H)}\right] \qquad (120)$$

$$b = \frac{1}{\sqrt{2}}\left[2(\rho_b K_S - \rho_o K_B)e^{S_2(-H)} - (\rho_b K_S + \rho_o K_B)e^{-S_2(-H)}\right] . \qquad (121)$$

Dividing, we have

$$\theta_L = \text{Arctan}\left[\frac{2D_2 + 1}{2D_2 - 1}\right] , \qquad (122)$$

where D_2 is defined as in equation (111). Note that this equation also does not approach the bottom reflected boundary condition (equation 84) as H approaches zero (the lower turning point approaches the bottom).

Lauvstad (1971) derived an expression for the phase coefficients of a wave entering a barrier region in terms of the phase of the wave leaving the opposite side of the barrier:

$$a_1 = \frac{1}{\sqrt{2}}\left[(a_2 + b_2)e^{-L} + (a_2 - b_2)e^{L}\right]$$

$$b_1 = \frac{1}{\sqrt{2}}\left[(a_2 + b_2)e^{-L} - (a_2 - b_2)e^{L}\right] . \qquad (123)$$

By rearranging, we obtain

$$a_2 = \frac{\sqrt{2}}{4}\left[(a_1 + b_1)e^{L} + (a_1 - b_1)e^{-L}\right]$$

$$b_2 = \frac{\sqrt{2}}{4}\left[(a_1 + b_1)e^{L} - (a_1 - b_1)e^{-L}\right], \tag{124}$$

which agrees with equation (114), the NORMO4 result,

$$\theta_L = \text{Arctan}\left[\frac{(a_1 + b_1)e^{L} + (a_1 - b_1)e^{-L}}{(a_1 + b_1)e^{L} - (a_1 - b_1)e^{-L}}\right]. \tag{114}$$

APPENDIX D

INPUT DATA

The following format is used for the input data to all three programs, NORMO1, NORMO4 and NORMO5.

VARIABLE	FORMAT	MEANING	RANGE
CARD ONE			
HED (20)	20A4	HEADING. TITLE CARD	
CARD TWO			
NUMV	I10	Number of Depth/Sound Speed Pairs. Negative if conversion from feet to meters desired.	2 - 50
NMOD	I10	Number of Modes Desired	1 - 100
IEX	I10	Iteration limit.	1 - 14
N	I10	Number of grid points desired	100 - 500 100 - 1000*
IOUT	I10	Type of output desired	see below
IDEV*	I2	Output device for card file	
CARD THREE			
H	F10.3	Bottom Depth	
CB	F10.3	Bottom Sound Speed	
DRO	F10.3	Water density	
DRB	F10.3	Bottom density	
HPLOT	F10.3	Maximum depth to be plotted	HPLOT H

*NORMO4 and NORMO5 only.

CARD FOUR

NFREQ	I10	Number of frequencies desired	1 - 20

CARD FIVE (SIX)

FREQ (NFREQ)	8F10.3	Frequencies desired	

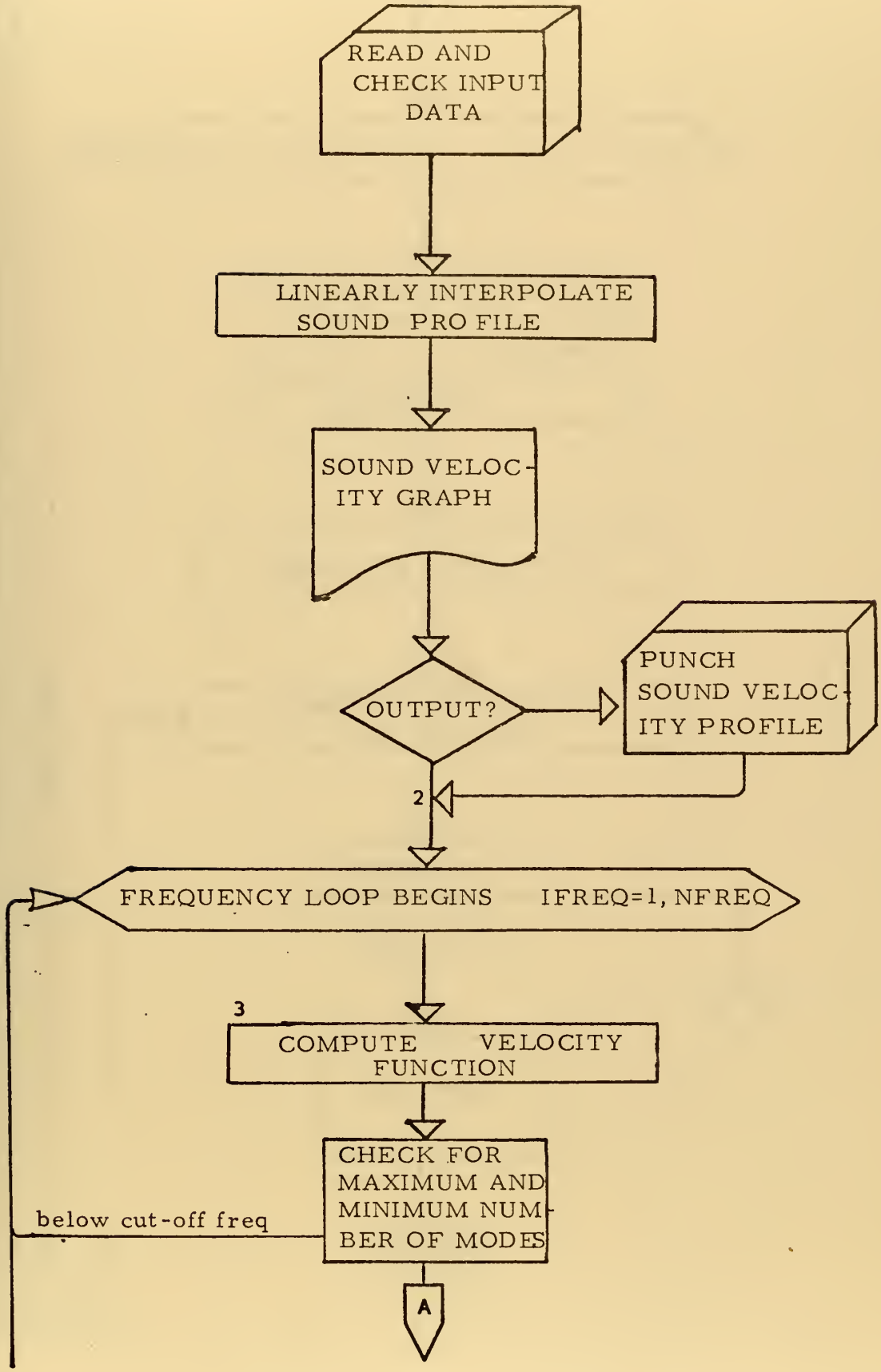
CARDS SIX TO 5+NUMV (SEVEN to 6+NUMV)

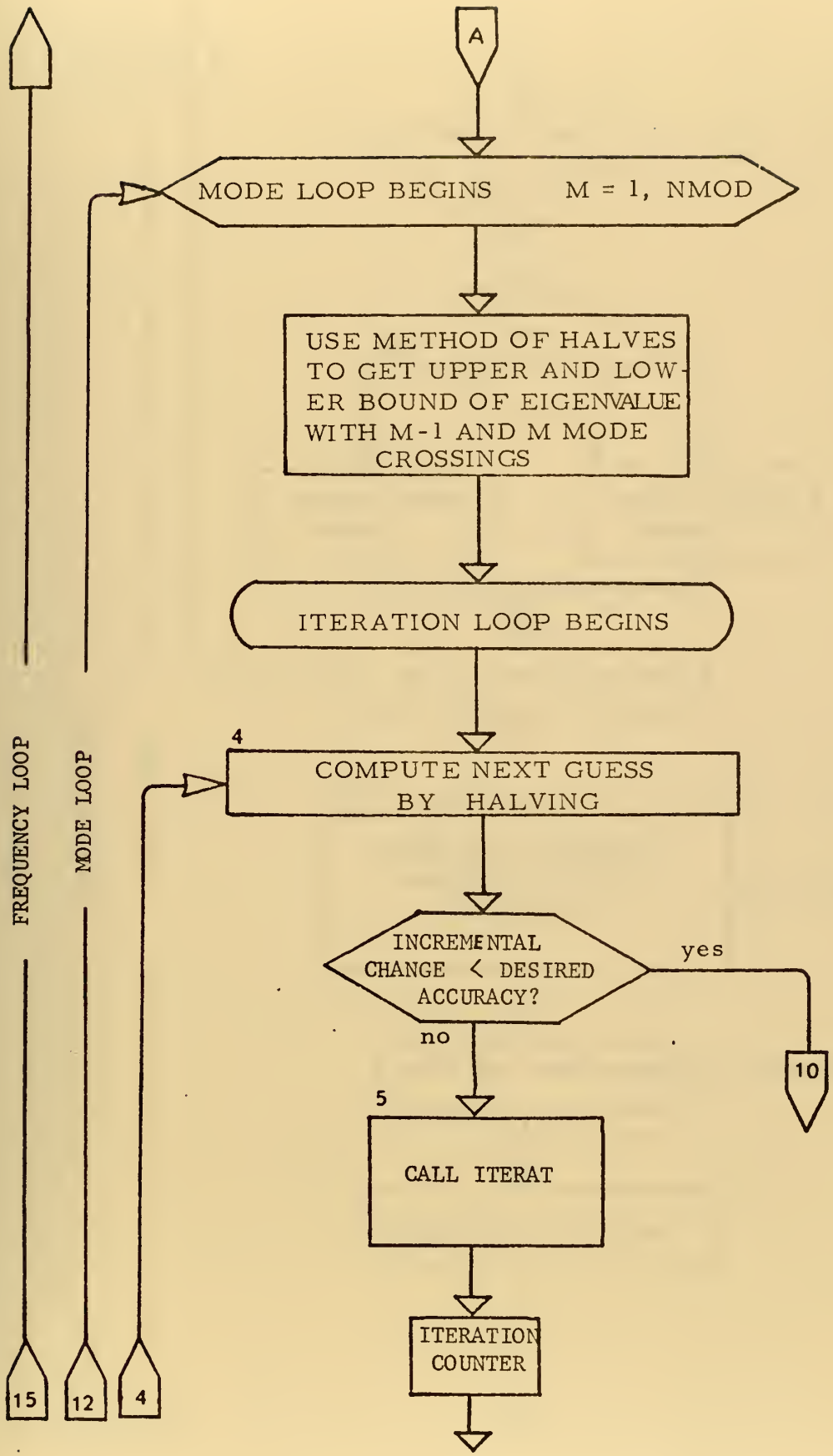
DEP(I)	F10.3	Depth	DEP(I) > DEP(I-1)
CC (I)	F10.3	Sound Speed	

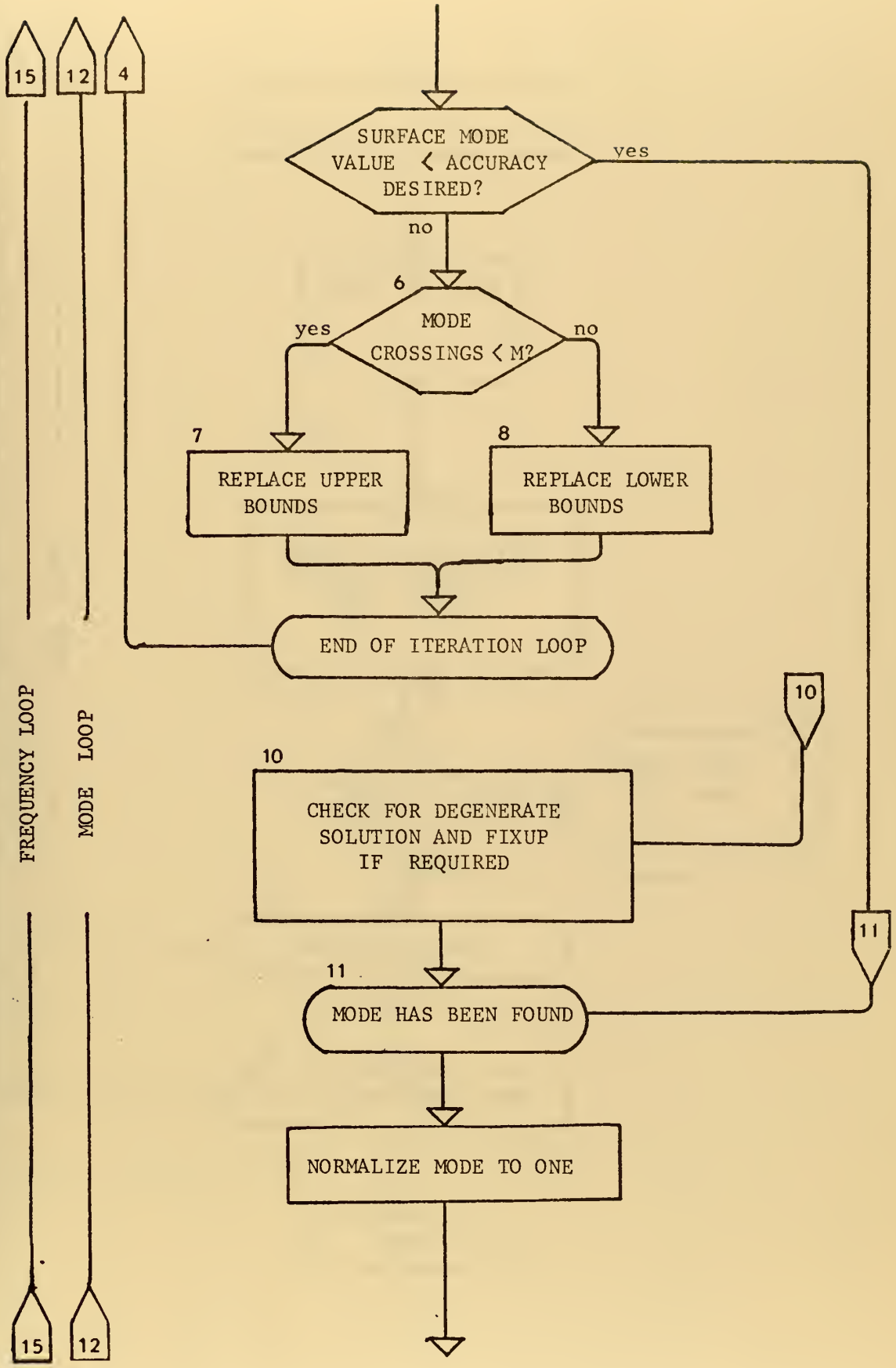
For NORMO1, IOUT=1 gives a punched deck output of the sound profile and mode parameters and profile. For NORMO4 and NORMO5, the IOUT parameter gives the following types of output:

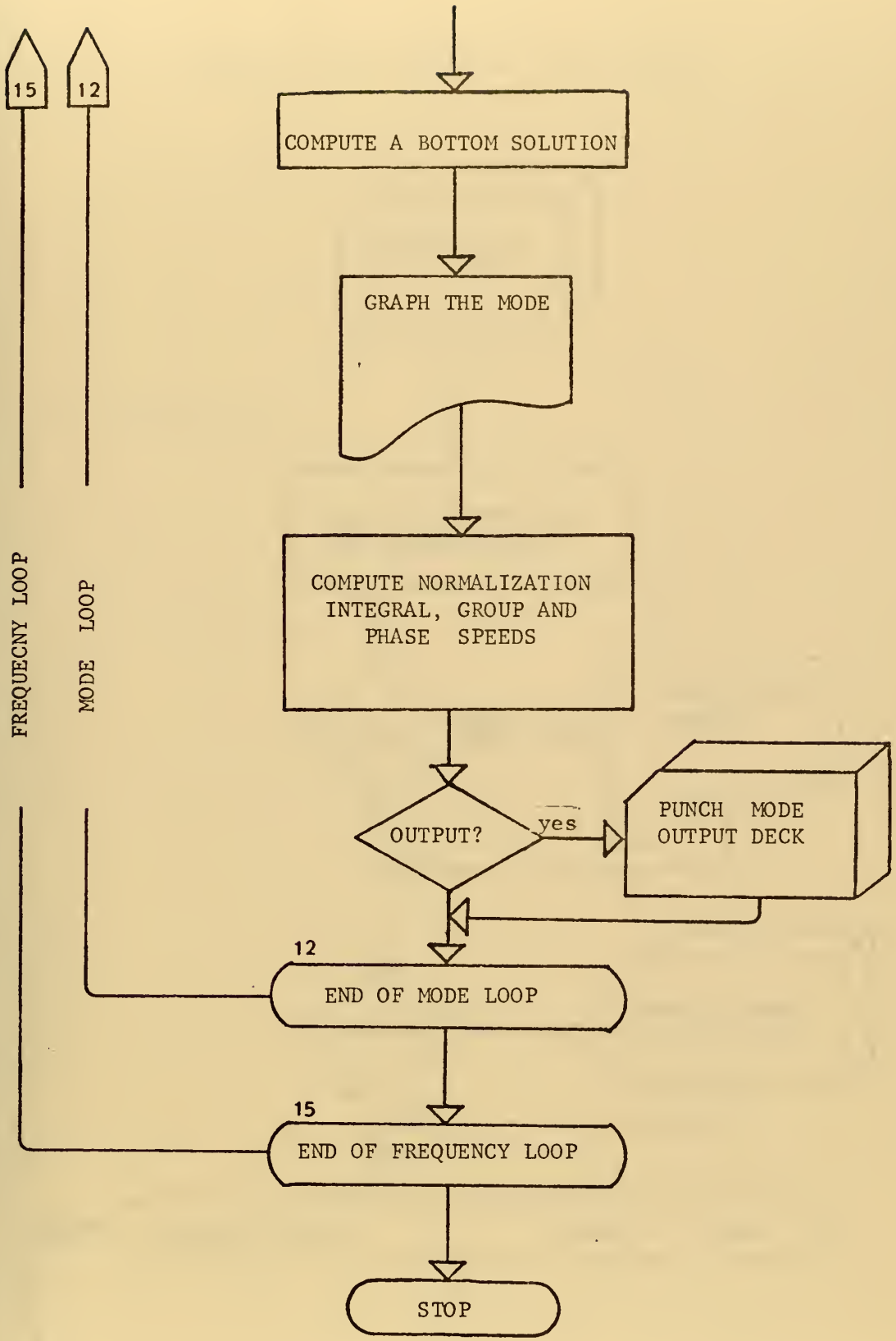
IOUT	PRINTER GRAPH	PUNCHED CARDS	DISK OUTPUT
0	YES	no	no
1	YES	YES	no
2	YES	no	YES
3	no	no	YES
4	no	YES	no
5	no	no	no

NORMO1 FLOWCHART

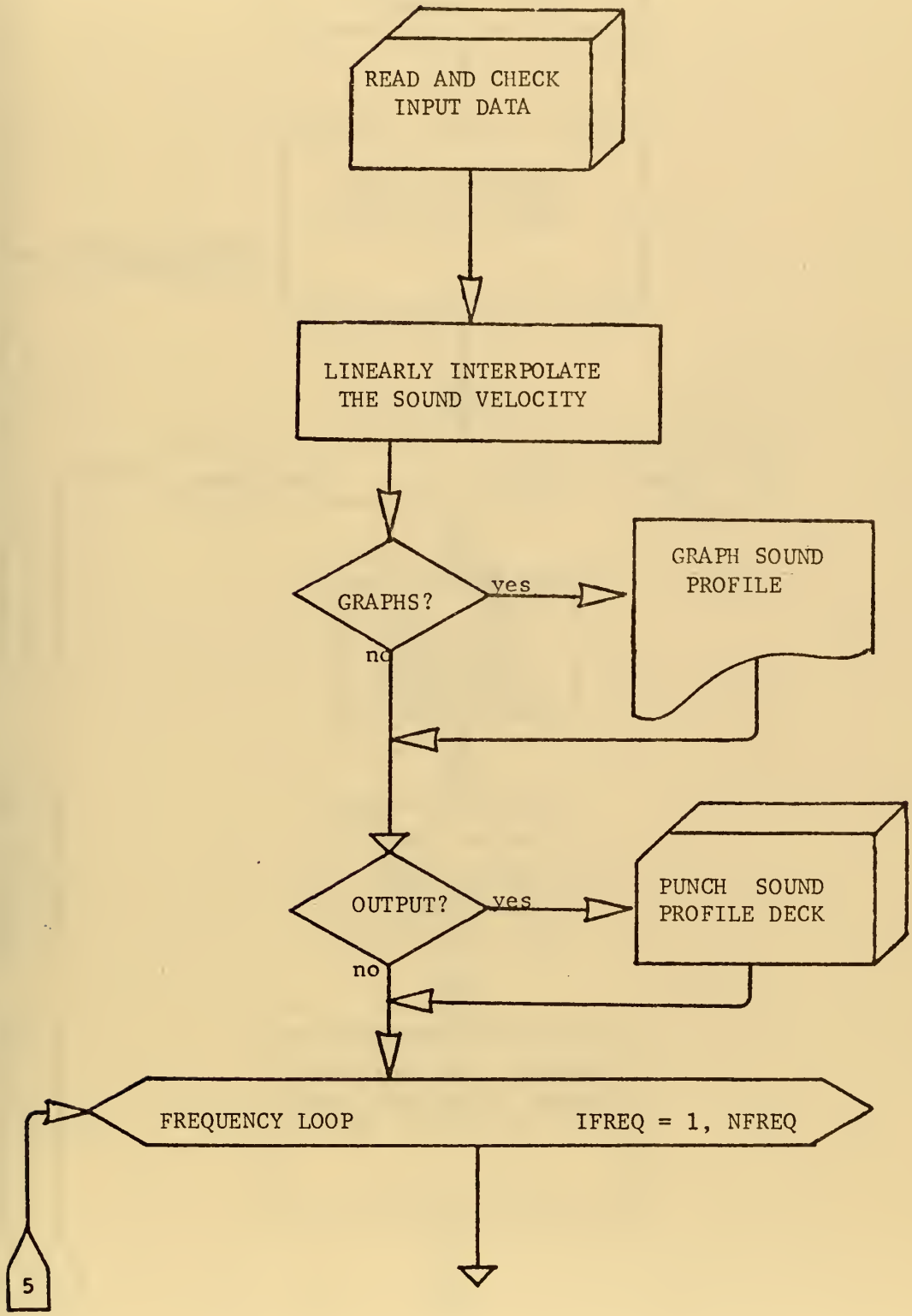


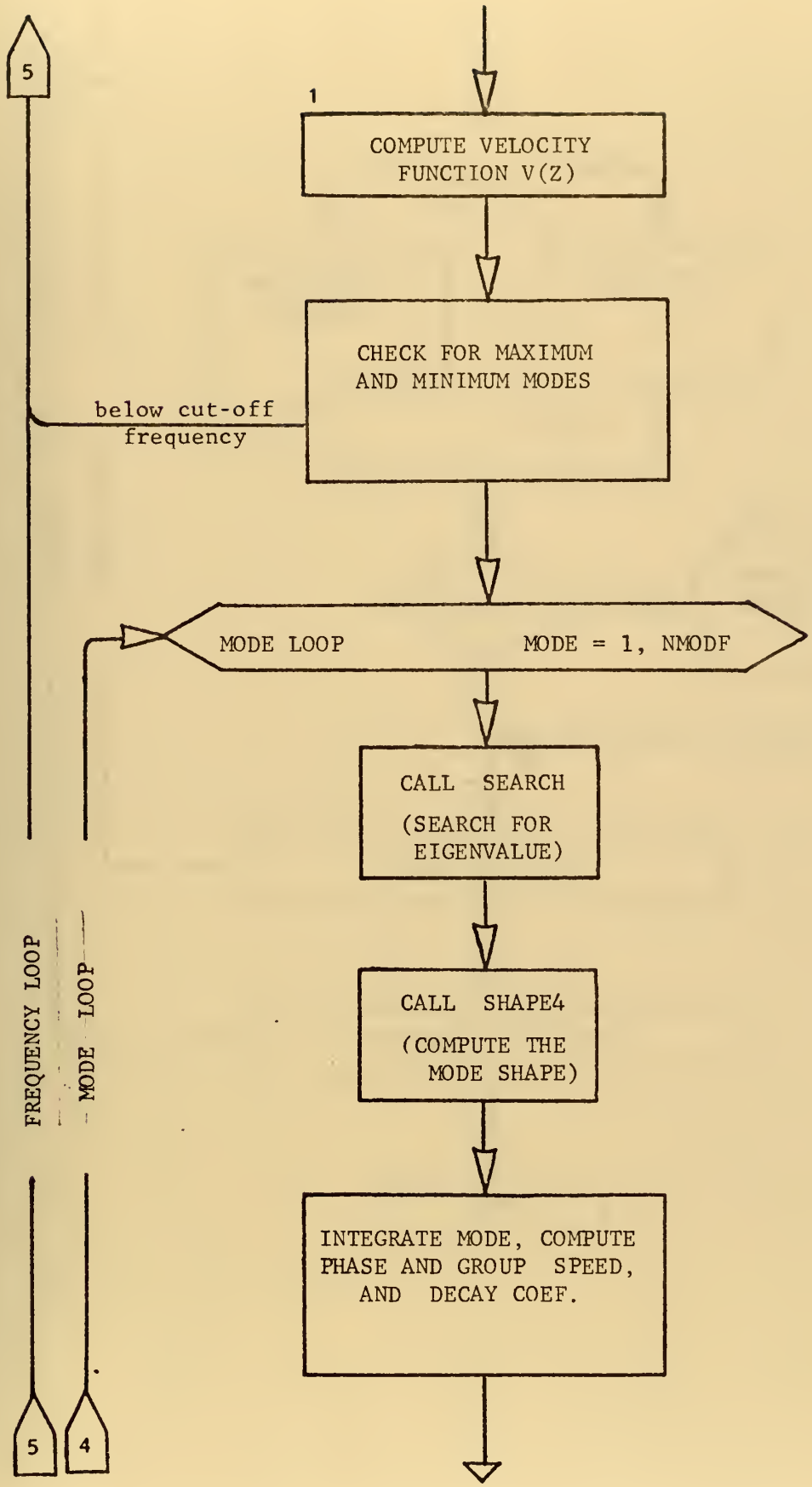

A
MODE LOOP BEGINS M = 1, NMOD
USE METHOD OF HALVES TO GET UPPER AND LOWER BOUND OF EIGENVALUE WITH M-1 AND M MODE CROSSINGS
ITERATION LOOP BEGINS
4
COMPUTE NEXT GUESS BY HALVING
INCREMENTAL CHANGE < DESIRED ACCURACY?
yes
no
10
5
CALL ITERAT
ITERATION COUNTER
FREQUENCY LOOP
MODE LOOP
15
12
4

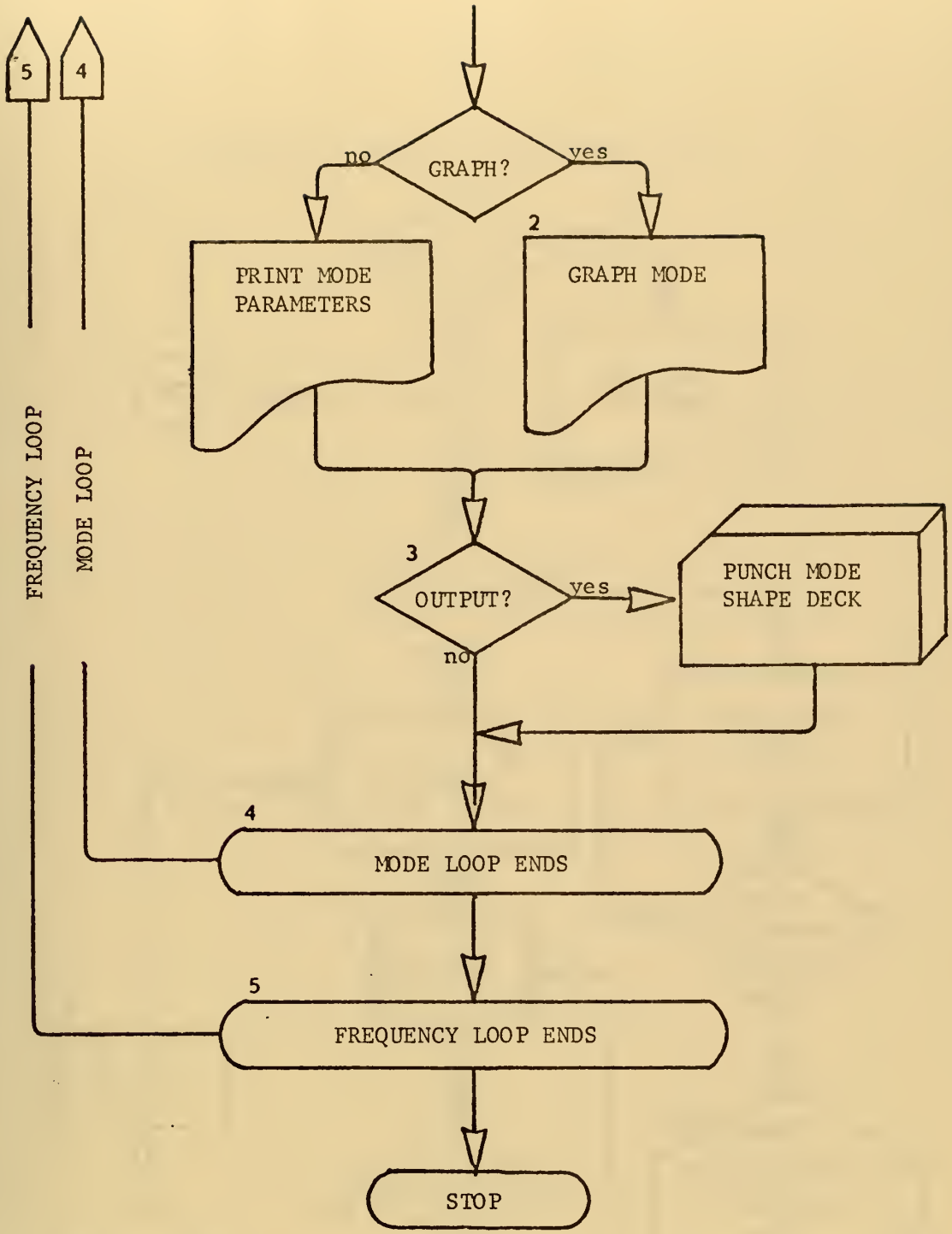

15
12
4
SURFACE MODE VALUE < ACCURACY DESIRED?
yes
no
6
MODE CROSSINGS < M?
yes
no
7
REPLACE UPPER BOUNDS
8
REPLACE LOWER BOUNDS
END OF ITERATION LOOP
10
10
CHECK FOR DEGENERATE SOLUTION AND FIXUP IF REQUIRED
11
11
MODE HAS BEEN FOUND
NORMALIZE MODE TO ONE
FREQUENCY LOOP
MODE LOOP
15
12


15
12
COMPUTE A BOTTOM SOLUTION
GRAPH THE MODE
COMPUTE NORMALIZATION INTEGRAL, GROUP AND PHASE SPEEDS
FREQUECNY LOOP
MODE LOOP
OUTPUT?
yes
PUNCH MODE OUTPUT DECK
12
END OF MODE LOOP
15
END OF FREQUENCY LOOP
STOP

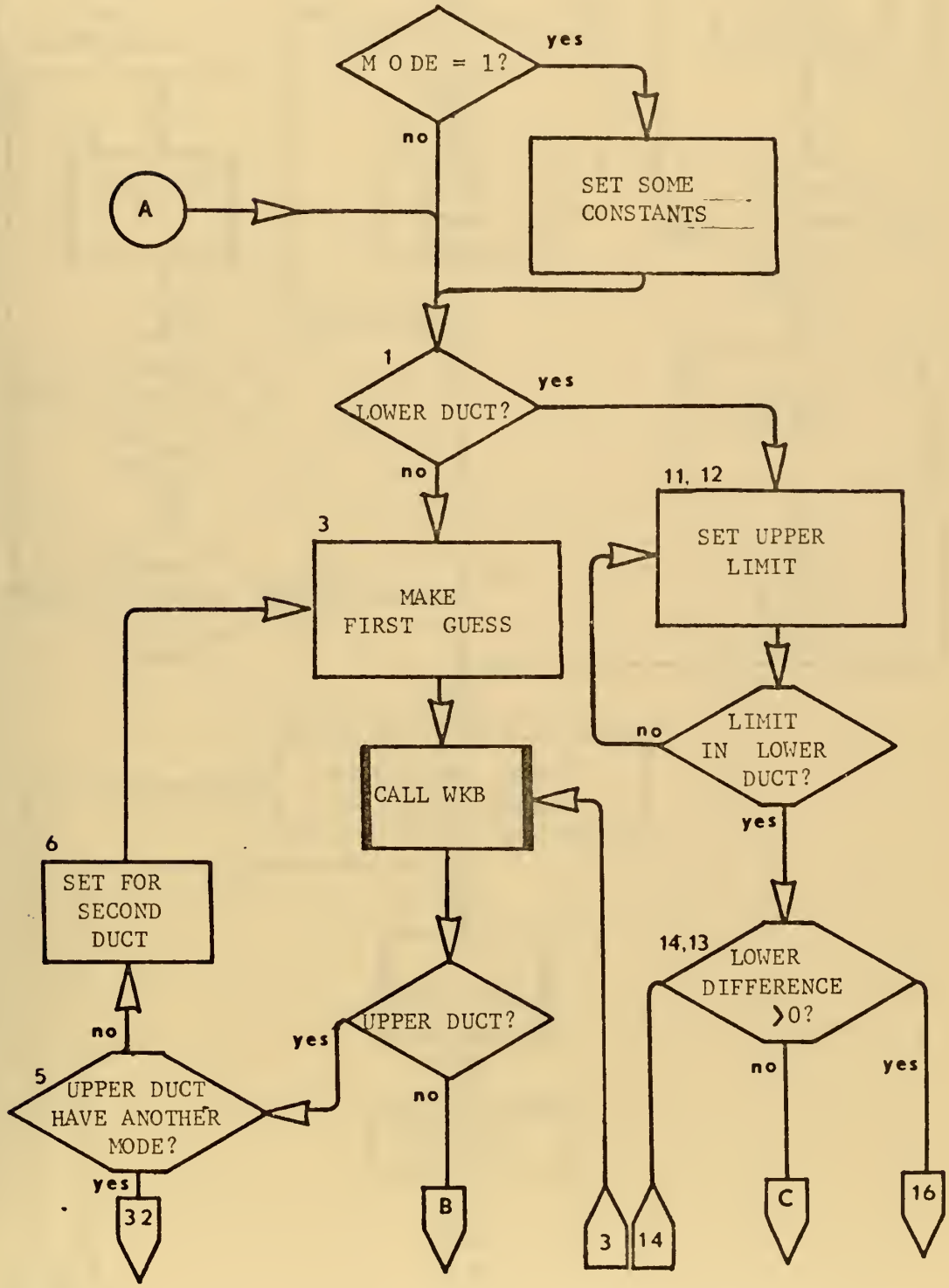
NORMO4 AND NORMO5 FLOWCHART

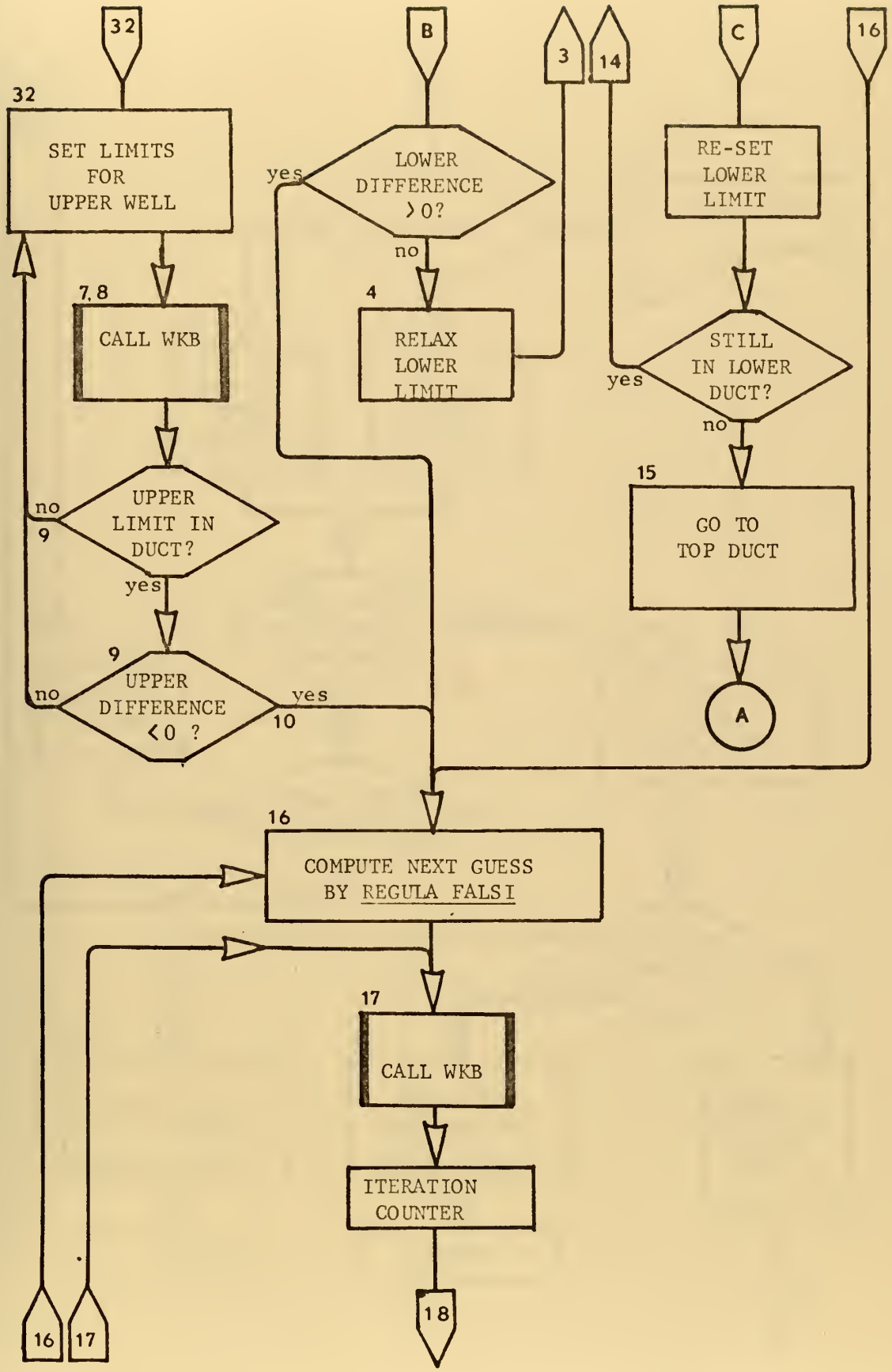


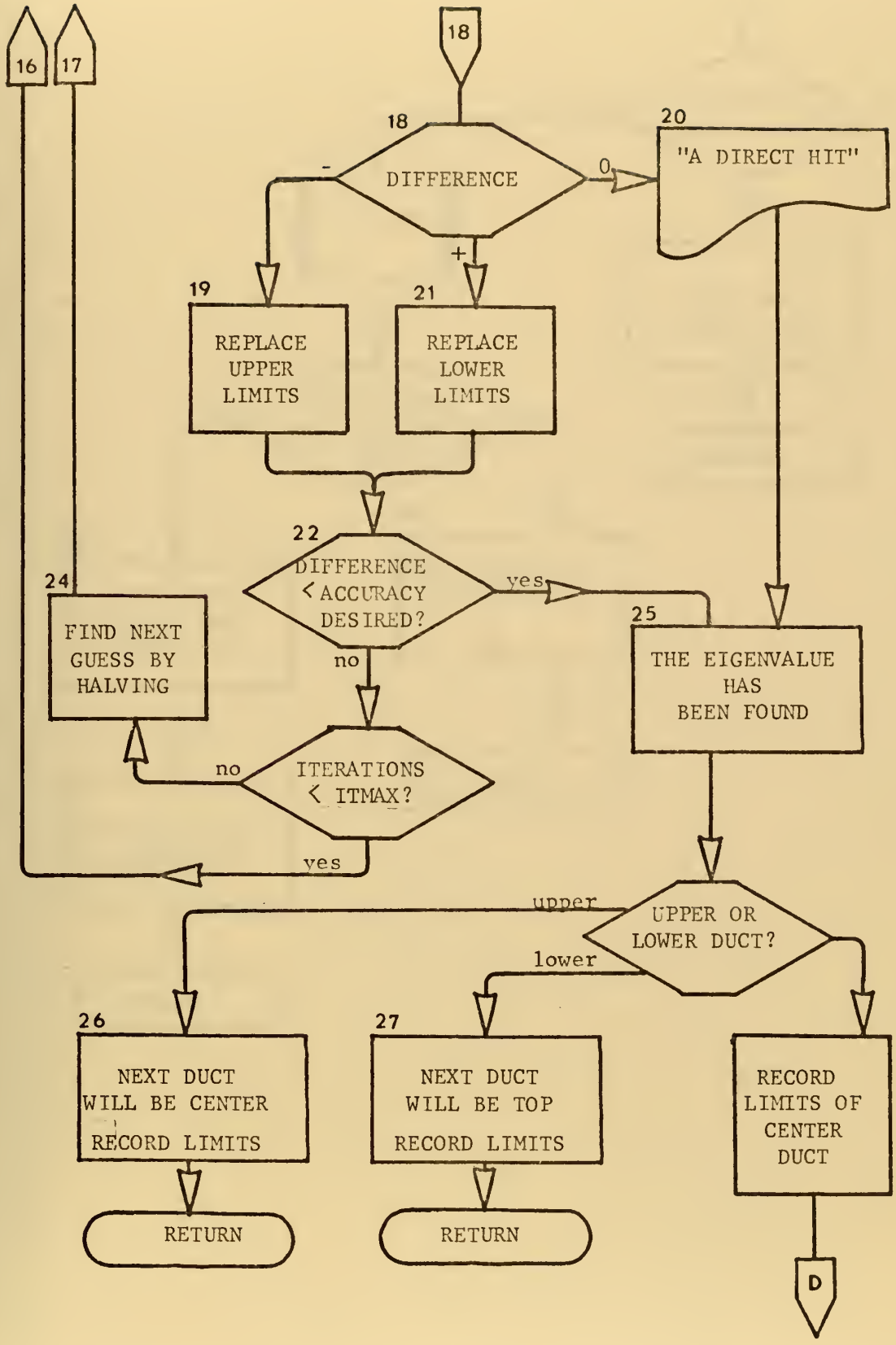

5
1
COMPUTE VELOCITY FUNCTION V(Z)
CHECK FOR MAXIMUM AND MINIMUM MODES
below cut-off frequency
MODE LOOP
MODE = 1, NMODF
CALL SEARCH
(SEARCH FOR EIGENVALUE)
CALL SHAPE4
(COMPUTE THE MODE SHAPE)
INTEGRATE MODE, COMPUTE PHASE AND GROUP SPEED, AND DECAY COEF.
FREQUENCY LOOP
MODE LOOP
5
4

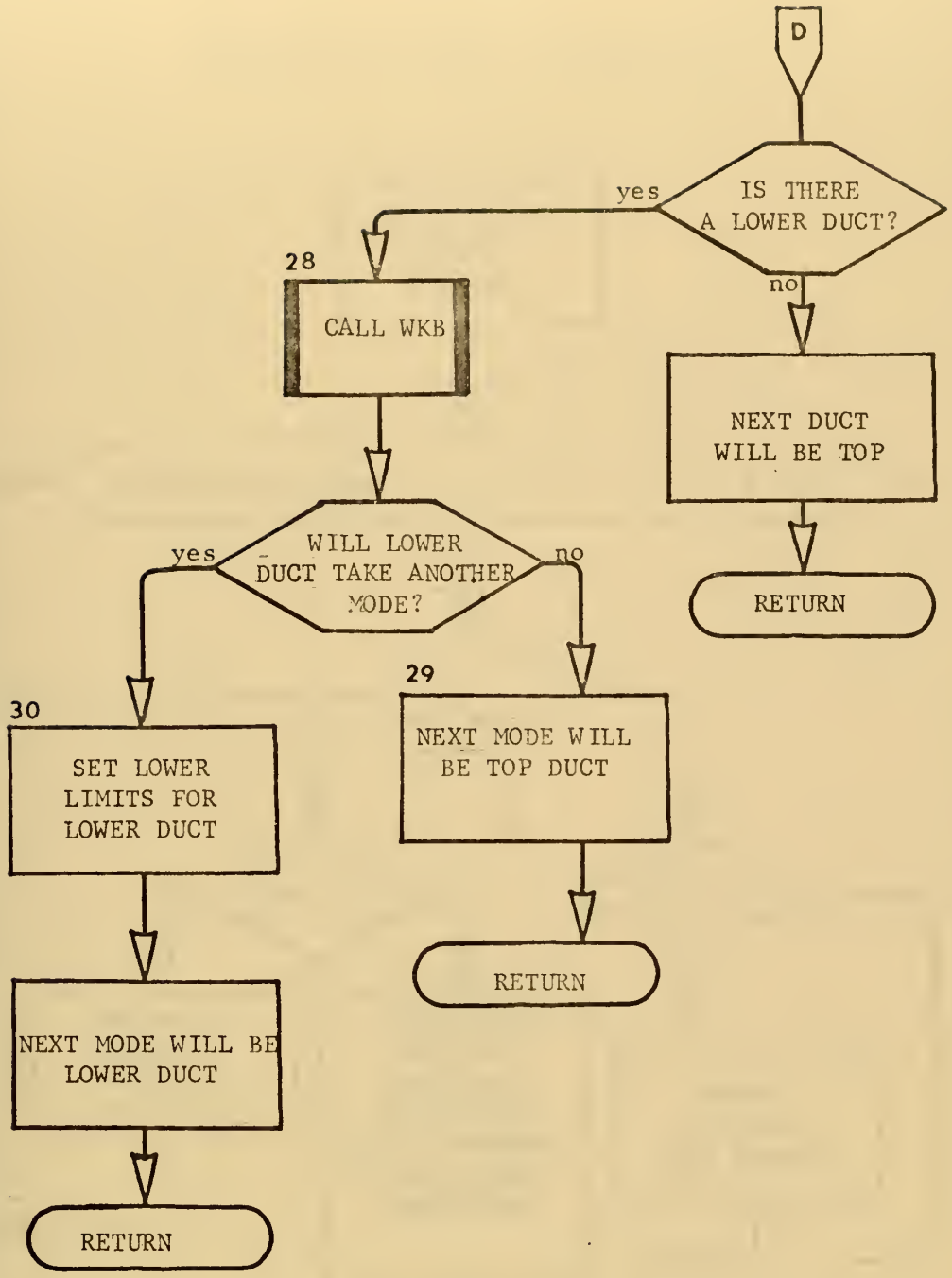

5
4
FREQUENCY LOOP
MODE LOOP
GRAPH?
no
yes
2
PRINT MODE PARAMETERS
GRAPH MODE
3
OUTPUT?
yes
no
PUNCH MODE SHAPE DECK
4
MODE LOOP ENDS
5
FREQUENCY LOOP ENDS
STOP

SEARCH SUBROUTINE FLOWCHART

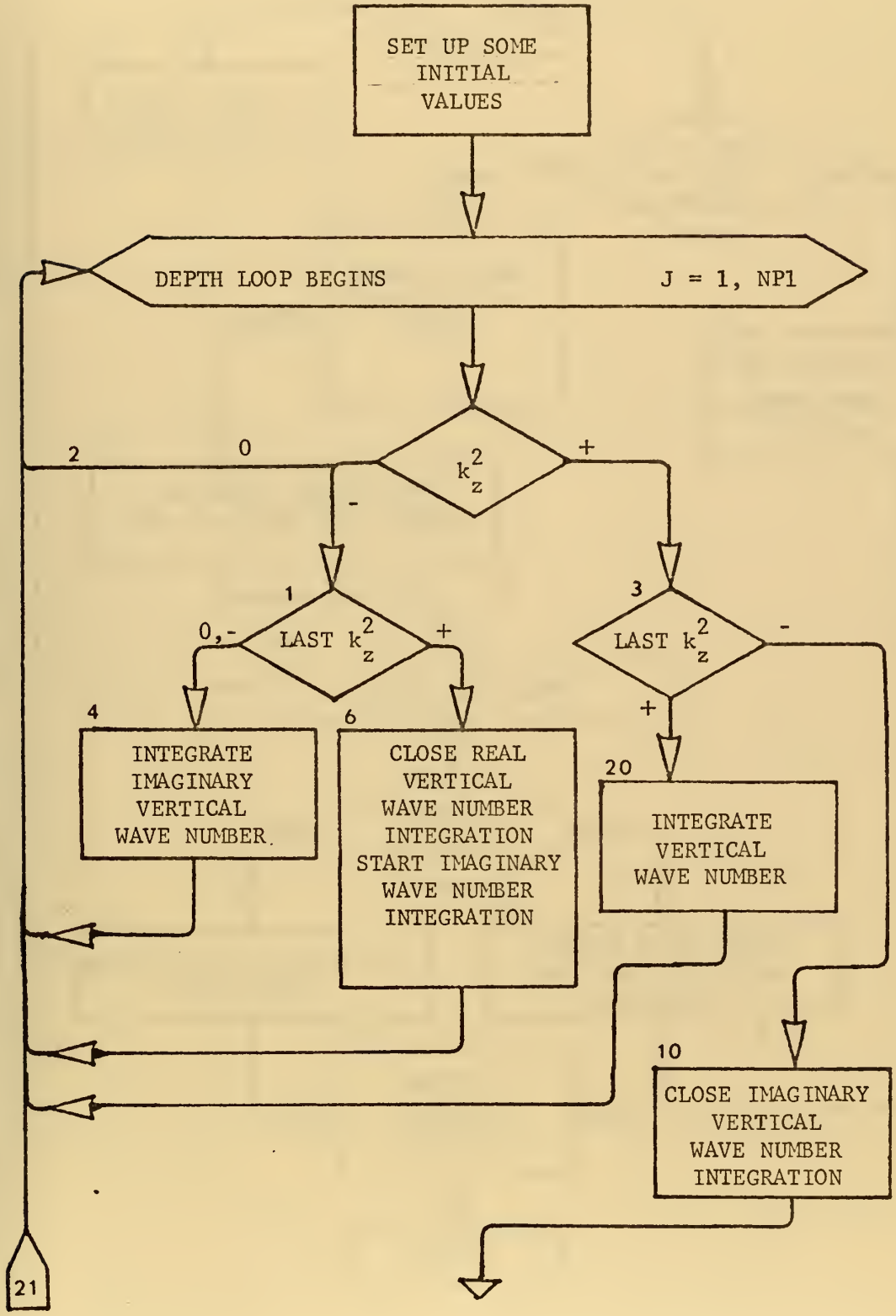


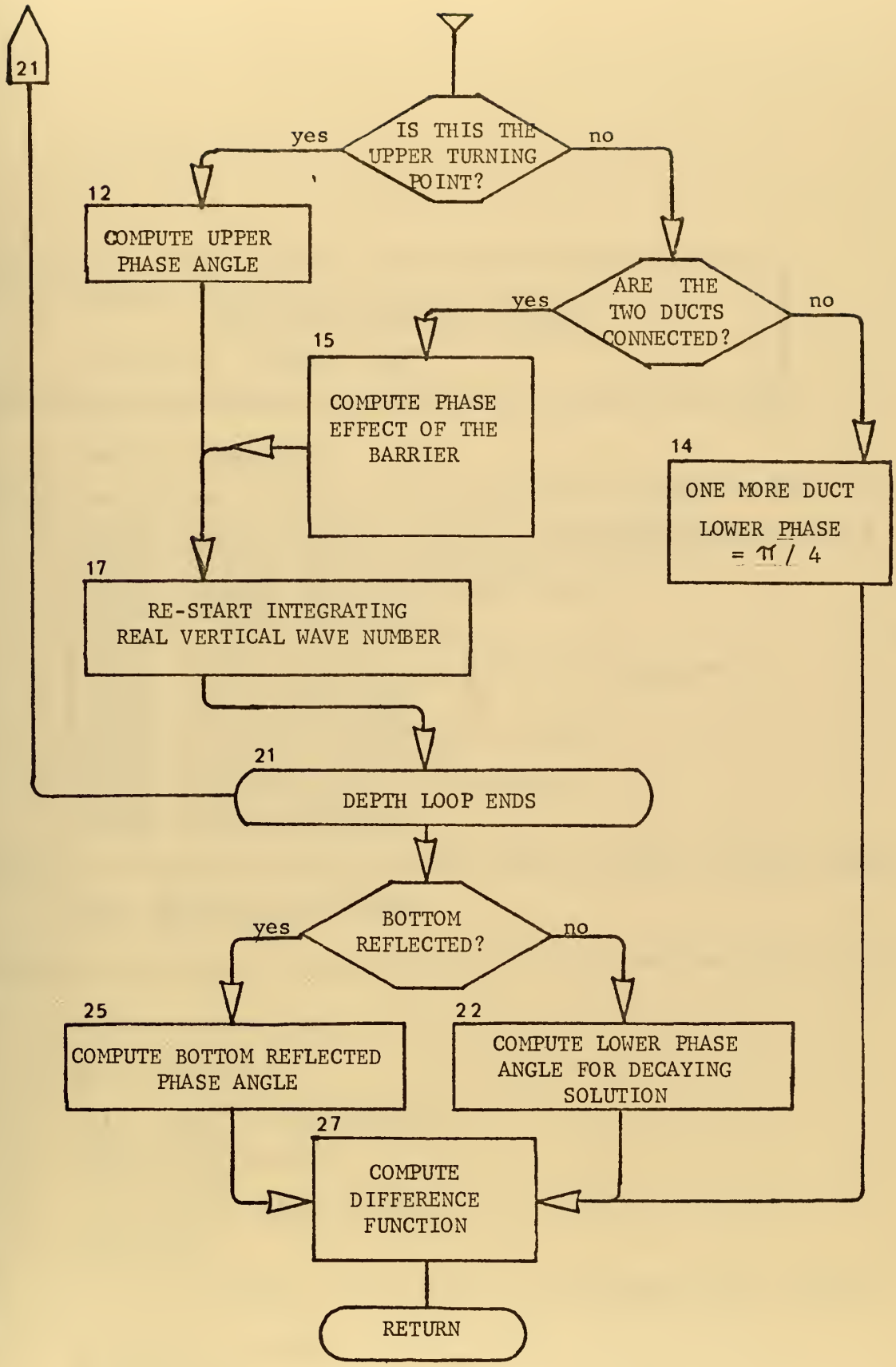

32
32
SET LIMITS FOR UPPER WELL
7,8
CALL WKB
9
UPPER LIMIT IN DUCT?
no
yes
9
UPPER DIFFERENCE <0 ?
no
yes
10
B
LOWER DIFFERENCE >0?
yes
no
4
RELAX LOWER LIMIT
3
14
C
RE-SET LOWER LIMIT
STILL IN LOWER DUCT?
yes
no
15
GO TO TOP DUCT
A
16
16
COMPUTE NEXT GUESS BY REGULA FALSI
17
CALL WKB
ITERATION COUNTER
18
16
17


16
17
18
18
DIFFERENCE
-
0
+
20
"A DIRECT HIT"
19
REPLACE UPPER LIMITS
21
REPLACE LOWER LIMITS
22
DIFFERENCE < ACCURACY DESIRED?
yes
no
24
FIND NEXT GUESS BY HALVING
no
ITERATIONS < ITMAX?
yes
25
THE EIGENVALUE HAS BEEN FOUND
upper
lower
UPPER OR LOWER DUCT?
26
NEXT DUCT WILL BE CENTER
RECORD LIMITS
27
NEXT DUCT WILL BE TOP
RECORD LIMITS
RECORD LIMITS OF CENTER DUCT
RETURN
RETURN
D


D
IS THERE A LOWER DUCT?
yes
no
28
CALL WKB
NEXT DUCT WILL BE TOP
RETURN
WILL LOWER DUCT TAKE ANOTHER MODE?
yes
no
30
SET LOWER LIMITS FOR LOWER DUCT
NEXT MODE WILL BE LOWER DUCT
RETURN
29
NEXT MODE WILL BE TOP DUCT
RETURN

WKB SUBROUTINE FLOWCHART




21
IS THIS THE UPPER TURNING POINT?
yes
no
12
COMPUTE UPPER PHASE ANGLE
ARE THE TWO DUCTS CONNECTED?
yes
no
15
COMPUTE PHASE EFFECT OF THE BARRIER
14
ONE MORE DUCT LOWER PHASE = π / 4
17
RE-START INTEGRATING REAL VERTICAL WAVE NUMBER
21
DEPTH LOOP ENDS
BOTTOM REFLECTED?
yes
no
25
COMPUTE BOTTOM REFLECTED PHASE ANGLE
22
COMPUTE LOWER PHASE ANGLE FOR DECAYING SOLUTION
27
COMPUTE DIFFERENCE FUNCTION
RETURN

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
C       NORMO1 : A NORMAL MODE EIGENVALUE                             C
C                AND EIGENFUNCTION SOLVER                             C
C                                                                     C
C       LT KIRK E. EVANS, USN                                         C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
      IMPLICIT REAL*8(D)
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< COMMON BLOCKS >>>>
C
      COMMON /DBL1/ DVZ(501),DH,DH2,DC6SQ,DRORB,DKAP,DUMAX
      COMMON /DBLE2/ DW2,DCB,DCMIN,DW,DC1,DC1SQ
      COMMON /DBLE3/ DUZ(501),DZZ(501)
      COMMON /DBLE4/ DS,DSSCC,DPVEL,DGVEL
      COMMON /PARAM/ N,NP1,IEX
      COMMON /SOUND/ Z(50),C(50)
      COMMON /HEAD/ HED(20)
      COMMON /SING1/ ZM,CB,CMIN,CMAX,PLTMAX,UM,FC
      COMMON /INPUT/ NUMV,NMOD
      COMMON /DMAP/ DK1(100)
      COMMON /DKMAP/ DKMIN,DKMAX,DKL,DKU
      COMMON /DSOUND/ DCC(501)
      COMMON /RHC/ RO,RB
      COMMON /BOTTO/ DECAY,ZB(20),UB(20)
      COMMON /FREQ/ FQV(20)
      COMMON /ACC/ DEPS,DIFFI
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< DATA >>>>>>>>>>>>>
C
      DATA DP2/6.283185306D0/
      DATA NPRINT/6/
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< PROGRAM BEGINS >>>
C
C
      WRITE (NPRINT,16)
C
      DO 1 I=1,100
    1 DK1(I)=-1.CD0
C
      CALL INPUT1 (IOUT,NFREQ)
      CALL INTRPL
      DCMIN2=DCMIN*DCMIN
      CALL CZPLOT
      IF (IOUT.NE.1) GO TO 2
      CALL PROFIL (DZZ,DCC,NP1)
    2 CONTINUE
      NMODP=NMOD
C
C     ------------------------------ FREQUENCY LOOP BEGINS
C
C
      DO 15 IF=1,NFREQ
      FC=FQV(IF)
      DW=DP2*FC
      DW2=DW*DW
```

```
      DKAP=DW/DCMIN
C
C-----------------------------COMPUTE THE V(Z) FUNCTION
C
C
      DO 3
      DCZ=DCC(J)
      DCZ2=DCZ*DCZ
    3 DVZ(J)=DW2*((DCZ-DCMIN)*(DCZ+DCMIN))/(DCZ2*DCMIN2)
C
C
C
      WRITE (NPRINT,18) FQ
      NMOD=NMODP
      CALL MAXMIN (&13)
C
C     ------------------------------ MODE LOOP BEGINS
C
C
      DO 12 M=1,NMOD
      IT=1
      CALL HALF (M,&9)
C
C     ------------------------ ITERATION LOOP BEGINS
C
    4 DKN=(DKL+DKU)*0.5D0
      DINC=DABS(DKU-DKL)*0.5DC/DKN
      IF (DINC-DIFFI) 10,5,5
C
    5 CALL ITERAT (DKN,MS)
      IT=IT+1
      DEPSIL=DUMAX*DEPS
      IF (DABS(DUZ(NP1))-DEPSIL) 11,6,6
C
    6 IF (MS-M) 7,8,8
C
    7 DKU=DKN
      GO TO 4
C
    8 DKL=DKN
      GO TO 4
C
C     ------------------------  MODE HAS BEEN FOUND
C
    9 DKN=(DKL+DKU)*0.5D0
C
   10 CALL FIXUP (DKN)
C
   11 DKMAX=DKN
      DK1(M)=DKN
C
      CALL DNORM1 (DUZ,DUMAX,NP1)
      CALL BOTTOM (DKN,M)
      CALL MODPLT (M,1)
      WRITE (NPRINT,17) IT
      CALL INTEGR (DKN,M)
      IF (IOUT.NE.1) GO TO 12
      CALL OUTPUT (DUZ,DKN,M,NP1)
   12 CONTINUE
C
C
C     ------------------------------ MODE LOOP ENDS
C
C
   13 DO 14 II=1,100
   14 DK1(II)=-1.0D0
C
   15 CONTINUE
C
C
C     ---------------------------- FREQUENCY LOOP ENDS
C
```

```
      STOP
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< FORMAT STATEMENTS >>
C
C
C
   16 FORMAT ('1  K.E. EVANS NORMAL MODE EIGENVALUE',
     *'SOLVER, VERSION 1, REVISION 6, DATED 13 AUGUST 73')
   17 FORMAT ('0   NUMBER OF ITERATIONS =', I5)
   18 FORMAT ('0    FREQUENCY : ', F6.1, ' HZ.')
      END
```

```
      SUBROUTINE INPUT1 (IOUT,NFREQ)
      IMPLICIT REAL*8(D)
C     ***************************************************************
C     *
C     *    THIS SUBROUTINE INPUTS THE SOUND AND RUN DATA,
C     *    DISPLAYS IT, THEN SETS SOME CONSTANTS FOR
C     *    LATER USE.
C     *
C     ***************************************************************
C
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< COMMON BLOCKS >>>>>>>>>>>
C
      COMMON /DBL1/ DVZ(501),DH,DH2,DCBSQ,DRORB,DKAP,DUMAX
      COMMON /DBLE2/ DW2,DCB,DCMIN,DW,DC1,DC1SQ
      COMMON /HEAD/ HED(20)
      COMMON /SING1/ ZM,CB,CMIN,CMAX,PLTMAX,UM,FC
      COMMON /SOUND/ Z(50),C(50)
      COMMON /INPUT/ NUMV,NMOD
      COMMON /PARAM/ N,NP1,IEX
      COMMON /DHHH/ DH2SIX,DH2D3,DHHLF
      COMMON /RHO/ RO,RB
      COMMON /FREQ/ FQV(20)
      COMMON /ACC/ DEPS,DIFFI
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< DATA >>>>>>>>>>>>>>>
C
      DATA NREAD,NPRINT/5,6/
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< PROGRAM BEGINS >>>>>
C
C
      CONVRT=1.0E0
C
CI-------------------------READ IN THE RUN PARAMETERS
C
      READ (NREAD,4) HED
      WRITE (NPRINT,5) HED
C
      READ (NREAD,6) NUMV,NMOD,IEX,N,ICUT
      N=IABS(N)
      IF (N.GT.500) N=500
      IF (N.LT.50) N=100
      NMOD=IABS(NMOD)
      IF (NMOD.GT.100) NMOD=100
      IF (NMOD.EQ.0) NMOD=10
      IF (N.EQ.0) N=500
      IEX=IABS(IEX)
      IF (IEX.GT.14) IEX=14
      IF (IEX.EQ.0) IEX=7
      NP1=N+1
      READ (NREAD,7) ZM,CB,RO,RB,PLTMAX
      ZM=ABS(ZM)
      PLTMAX=ABS(PLTMAX)
      READ (NREAD,6) NFREQ
      NFREQ=IABS(NFREQ)
      IF (NFREQ.GT.20) NFREQ=20
      READ (NREAD,7) (FQV(I),I=1,NFREQ)
      FC=FQV(1)
C
C-------------------------CONVERT FROM FEET TO METERS
C                                   IF REQUIRED
C
      IF (NUMV) 1,2,2
    1 CONVRT=0.3048
      NUMV=-NUMV
      ZM=ZM*CONVRT
      CB=CB*CONVRT
      PLTMAX=PLTMAX*CONVRT
C
    2 WRITE (NPRINT,8)
      WRITE (NPRINT,9) (FQV(I),I=1,NFREQ)
```

```
      WRITE (NPRINT,10) NMOD,IEX
      WRITE (NPRINT,11) ZM,PLTMAX
      WRITE (NPRINT,12) RO,RB
      WRITE (NPRINT,13) CB,N
C
C
C-----------------------------READ IN ANC DISPLAY SCUND DATA
C
      WRITE (NPRINT,14) HED
C
C
      DC 3 I=1,NUMV
      READ (NREAC,7) Z(I),CI
      CI=CI*CCNVRT
      Z(I)=Z(I)*CONVRT
      C(I)=CI
    3 WRITE (NPRINT,15) Z(I),CI
C
C
C-----------------------------COMPUTE SCME CONSTANTS
C
      DH=DBLE(ZM)/DFLOAT(N)
      DZM=DBLE(ZM)
      DRCRB=DBLE(RO/RB)
      DH2=DH*DH
      DCB=DBLE(CB)
      DCBSQ=DCB*DCB
      DH2SIX=DH2*0.16666666666666667D0
      DH2D3=DH2*0.33333333333333333D0
      DHHLF=DH*0.5D0
      DEPS=1.0D1**(-IEX)
      DIFFI=DEPS*DEPS
      DIFFI=DMAX1(DIFFI,1.0D-16)
C
      RETURN
C
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< FORMAT STATEMENTS >>
C
C
C
    4 FCRMAT (20A4)
    5 FCRMAT ('1',20A4)
    6 FCRMAT (5I10)
    7 FCRMAT (8F10.3)
    8 FCRMAT ('0', T34, 'RUN PARAMETERS')
    9 FCRMAT ('0',T30,'FRECUENCIES =', (T42,F6.1,' HZ',/))
   10 FCRMAT ('0',T24,  'MODES REQUESTEC :', I4, //,
     1               T24,'ITERATION LIMIT = 10**( -',I2,')')
   11 FCRMAT ('C',T27, 'BCTTOM DEPTH :',F8.1,' METERS',//,
     1               T22, 'MAX DEPTH PLOTTEC :',F8.1,' METERS')
   12 FCRMAT ('0',T26, 'CCEAN DENSITY :', F7.4,
     *       ' GM/CM**3', //,
     1               T25, 'BOTTOM DENSITY :',F7.4,' GM/CM**3')
   13 FORMAT ('0',T18, 'BCTTCM SOUND VELCCITY :',
     *      F8.1,' METERS/SEC',//,
     1               T17, 'NUMBER OF GRIC PCINTS :', I5)
   14 FCRMAT ('1', 20A4, //, T33, 'VELCCITY PRCFILE',//,T22,
     1           'DEPTH, METERS',T44,'SOUND VELOCITY,M/SEC',/)
   15 FCRMAT ('0', T25, F6.1, T57, F6.1 )
      END
```

```
      SUBROUTINE INTRPL
      IMPLICIT REAL*8(D)
C
CC    ***********************************************************
CC    *
CC    *   THIS SUBROUTINE INTERPOLATES THE DEPTH AND SOUND
C     *   VELOCITY DATA GIVEN, FOR THE GRID VALUES.
C     *   IN ADDITION, IT COMPUTES SOME CONSTANTS FOR
C     *   LATER USE, AND EXTRAPOLATES THE SOUND VELOCITY
C     *   TO THE BOTTOM IF REQUIRED.
C     *
C     ***********************************************************
      COMMON /DBL1/ DVZ(501),DH,DH2,DCBSQ,DRORB,DKAP2,DUMAX
      COMMON /DBLE2/ DW2,DCB,DCMIN,DW,DC1,DC1SQ
      COMMON /DBLE3/ DUZ(501),DZZ(501)
      COMMON /PARAM/ N,NP1,IEX
      COMMON /SING1/ ZM,CB,CMIN,CMAX,PLTMAX,UM,FC
      COMMON /DSOUND/ DCC(501)
      COMMON /SOUND/ Z(50),C(50)
      COMMON /INPUT/ NUMV,NMOD
C
CC
      CMAX=0.0
      CMIN=CB
C
CC    *** EXTRAPOLATE THE PROFILE TO THE BOTTOM  ***
C
      IF (Z(NUMV)-ZM) 1,3,2
C
    1 NUMV=NUMV+1
C
    2 C(NUMV)=C(NUMV-1)+(ZM-Z(NUMV-1))*0.17E-1
      Z(NUMV)=ZM
C
    3 NUMVP1=NUMV+1
      NUMVD2=NUMV/2
C
CC
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< SWAPPING AND TESTING LOOP
C
      DO 4 I=1,NUMVD2
      NUP=NUMVP1-I
      CL=C(I)
      CU=C(NUP)
      ZL=Z(I)
      ZU=Z(NUP)
      CMAX=AMAX1(CMAX,CL,CU)
      CMIN=AMIN1(CMIN,CL,CU)
C
      C(I)=CU
      C(NUP)=CL
      Z(I)=ZM-ZU
      Z(NUP)=ZM-ZL
    4 CONTINUE
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< SWAPPING AND TESTING LOOP ENDS
C
      IF (NUMVD2*2.EQ.NUMV) GO TO 5
      CNVD21=C(NUMVD2+1)
      CMAX=AMAX1(CMAX,CNVD21)
      CMIN=AMIN1(CMIN,CNVD21)
      Z(NUMVD2+1)=ZM-Z(NUMVD2+1)
C
CC    ***  SET UP FOR THE FINE GRID COMPUTATION  ***
C
    5 DCMIN=DBLE(CMIN)
      DCMIN2=DCMIN*DCMIN
      DC1=DBLE(C(1))
      DC1SQ=DC1*DC1
      DZM=DBLE(ZM)
      I=1
      DCI=DC1
```

```
      DCIP1=DBLE(C(2))
      DZI=DBLE(Z(1))
      DZIP1=DBLE(Z(2))
      DZDIFF=DZIP1-DZI
      DFLN=DFLOAT(N)
      DZP=0.0D0
C
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< FINE GRID LOOP BEGINS
C
      DO 8 J=1,NP1
      DZZ(J)=DZP
      IF (DZP-DZIP1) 7,7,6
C
C          ***  RESET THE I PARAMETERS  ***
C
    6 I=I+1
      DZI=DZIP1
      DCI=DCIP1
      DCIP1=DBLE(C(I+1))
      DZIP1=DBLE(Z(I+1))
      DZDIFF=DZIP1-DZI
C
C          ***  COMPUTE THE VALUES FOR C(Z)  ***
C
    7 DCZ=(DCI*(DZIP1-DZP)+DCIP1*(DZP-DZI))/DZDIFF
      DCC(J)=DCZ
C
C          ***  COMPUTE THE NEXT DEPTH  ***
C
      DZP=DZM*DFLOAT(J)/DFLN
    8 CONTINUE
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< FINE GRID LOOP ENDS
C
C
      RETURN
      END
```

```
      SUBROUTINE MAXMIN (*)
      IMPLICIT REAL*8(D)
C
C     ****************************************************************
C     *
C     *    THIS SUBROUTINE CHECKS THE MAXIMUM AND MINIMUM
C     *    VALUES OF THE HORIZONTAL WAVE NUMBER; THEN
C     *    DETERMINES THE NUMBER OF MODES  REPRESENTED
C     *    BY EACH.  IF NO MODES ARE PRESENT FOR THE
C     *    MINIMUM WAVE NUMBER THE FREQUENCY IS BELOW
C     *    CUT-OFF.  IF THE NUMBER OF MODES REQUESTED IS
C     *    ABOVE THAT REPRESENTED BY THE MINIMUM WAVE
C     *    NUMBER, THE REQUEST IS MODIFIED ACCORDINGLY.
C     *
C     ****************************************************************
C
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< COMMON BLOCKS >>>>>>
C
C
      COMMON /DBL1/ DVZ(501),DH,DH2,DCBSQ,DRORB,DKAP,DUMAX
      COMMON /DBLE2/ DW2,DCB,DCMIN,DW,DC1,DC1SQ
      COMMON /DBLE3/ DUZ(501),DZZ(501)
      COMMON /PARAM/ N,NP1,IEX
      COMMON /HEAD/ HED(20)
      COMMON /SING1/ ZM,CB,CMIN,CMAX,PLTMAX,UM,FC
      COMMON /DMAP/ DK1(100)
      COMMON /DKMAP/ DKMIN,DKMAX,DKL,DKU
      COMMON /INPUT/ NUMV,NMOD
      COMMON /SOUND/ Z(50),C(50)
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< DATA >>>>>>>>>>>>>>
C
      DATA NREAD,NPRINT/5,6/
      DATA MAXI,MINI/'MAXI', 'MINI'/
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< PROGRAM BEGINS >>>>>
C
C
C     ***  COMPUTE THE MAXIMUM AND MINIMUM DK'S   ***
C
      DKMIN=DW/DCB
      DKMAX=DW/DCMIN
C
C     ***  CHECK OUT THE  DKMIN FOR MAX NO. OF MODES   ***
C
      CALL ITERAT (DKMIN,MS)
      IF (MS) 1,4,1
    1 MODMAX=MS
      DK1(MS)=DKMIN
C
      MODMIN=0
      MS=0
C
C     ***  PRINT THE RESULTS   ***
C
      WRITE (NPRINT,5) HED
      WRITE (NPRINT,6)
      WRITE (NPRINT,7) MAXI,DKMAX,MODMIN
      WRITE (NPRINT,7) MINI,DKMIN,MODMAX
C
C     ***  CHECK THE  NUMBER OF MODES REQUESTED   ***
C
      IF (MODMAX-NMOD) 2,3,3
    2 NMOD=MODMAX
      WRITE (NPRINT,8)
      WRITE (NPRINT,9)
      WRITE (NPRINT,10)
      WRITE (NPRINT,9)
      WRITE (NPRINT,8)
      WRITE (NPRINT,11) NMOD
C
```

```
    3 RETURN
C
C     *** BELOW CUT-OFF FREQUENCY ***
C
    4 WRITE (NPRINT,8)
      WRITE (NPRINT,9)
      WRITE (NPRINT,10)
      WRITE (NPRINT,9)
      WRITE (NPRINT,8)
      WRITE (NPRINT,12)
      RETURN 1
C
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< FORMAT STATEMENTS >>
C
C
C
C
    5 FORMAT ('1', 20A4)
    6 FORMAT ('0', ///, T25, 'BOUNDS ON HORIZONTAL',
     1              ' WAVE NUMBER')
    7 FORMAT ('0', T22, A4, 'NUM K :', F12.8, 3X,
     1              'MODES :', I4)
    8 FORMAT (T33,13('*'))
    9 FORMAT (T33,1H*,11X,1H*)
   10 FORMAT (T33,'*  CAUTION  *')
   11 FORMAT ('0', T26, 'MODES REQUESTED RESET TO :', I4)
   12 FORMAT ('0', /, T25, 'FREQUENCY REQUESTED IS ',
     1    'BELOW CUT-OFF', //, T20, 'JOB TERMINATED OR ',
     2    'NEW FREQUENCY SELECTED.')
      END
```

```
      SUBROUTINE HALF (M,*)
      IMPLICIT REAL*8(D)
C
CCC   ***********************************************************
CCC   *
CCC   *    THIS SUBROUTINE SEARCHES FOR AN UPPER VALUE
CCC   *    OF DK WITH M-1 MODE CROSSINGS AND A LOWER
CCC   *    VALUE OF DK WITH M MODE CROSSINGS.  AT THE
CCC   *    SAME TIME IT FILLS IN A MAP OF  DK VS.
CCC   *    MODE CROSSINGS FOR LATER USE.
CCC   *
CCC   ***********************************************************
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<COMMON BLOCKS >>>>>>>>>>>>>>>
      COMMON /DBLE2/ DW2,DCB,DCMIN,DW,DC1,DC1SQ
      COMMON /PARAM/ N,NP1,IEX
      COMMON /DMAP/ DK1(100)
      COMMON /DKMAP/ DKMIN,DKMAX,DKL,DKU
      COMMON /ACC/ DEPS,DIFFI
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< PROGRAM BEGINS >>>>>>>>>>
C
      LOW=-1
      LUP=-1
C
CCC   *** CHECK THE MAP FOR  ALREADY COMPUTED GLESES  ***
CCC
C
      DO 2 I=M,100
      IF (DK1(I)) 1,1,3
    1 CONTINUE
    2 CONTINUE
C
CCC
C
      DKL=DKMIN
      GO TO 4
C
    3 DKL=DK1(I)
    4 DKU=DKMAX
C
CCC   ***  FIND & CHECK THE NEW MID DK VALUE ***
C
    5 DKN=(DKL+DKU)*0.5D0
C
CCC   ***  CHECK THE DIFFERENCE  **
C
      IF ((DKU-DKN)/DKN-DIFFI) 13,6,6
C
    6 CALL ITERAT (DKN,MS)
C
      IF (MS-M) 7,8,9
C
CCC   ***  IF WE GET TWO END POINTS WE ARE FINISHED  ***
CCC             ***  IF NOT, CONTINUE HALVING  ***
C
    7 DKU=DKN
      IF (LUP.GT.0) RETURN
      LOW=1
      GO TO 5
C
C
    8 DKL=DKN
      IF (LOW.GT.0) RETURN
      LUP=1
      GO TO 5
C
CCC   ***  FILL IN THE MAP IF NEEDED  ***
C
    9 IF (MS-100) 10,10,8
   10 IF (DKN-DK1(MS)) 12,12,11
```

```
   11 DK1(MS)=DKN
   12 DKL=DKN
      GO TO 5
C
C        ***  IF WE GET HERE WE HAVE COME TO WITHIN ***
C        ***  THE LIMITS OF DESIRED ACCURACY  IN DK ***
   13 DKL=DKN
      DKL=DKN
      RETURN 1
C
C
      END
```

```
      SUBROUTINE ITERAT (DK,MS)
      IMPLICIT REAL*8(D)
C
C
C     ******************************************************************
C     *
C     *
C     *   THIS SUBROUTINE ITERATES THE FINITE DIFFERENCE
C     *   EQUATION FROM THE BOTTOM TO THE SURFACE.  THE
C     *   COMPUTED VARIABLES ARE:
C     *
C     *       DUZ (I) : VALUE OF THE MODE EIGENFUNCTION
C     *       MS : NUMBER OF MODE CROSSINGS
C     *       DUMAX = ABSOLUTE MAXIMUM VALUE OF EIGENFUNCTIO
C     ******************************************************************
C
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< COMMON BLOCKS >>>>>>
C
C
      COMMON /DBL1/ DVZ(501),DH,DH2,DCBSQ,DRORB,DKAP,DUMAX
      COMMON /DBLE2/ DW2,DCB,DCMIN,DW,DC1,DC1SQ
      COMMON /DBLE3/ DUZ(501),DZZ(501)
      COMMON /PARAM/ N,NP1,IEX
      COMMON /SING1/ ZM,CB,CMIN,CMAX,PLTMAX,UM,FQ
      COMMON /DKMAP/ DKMIN,DKMAX,DKL,DKU
      COMMON /DHHH/ DH2SIX,DH2D3,DHHLF
C
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< DATA >>>>>>>>>>>>>>
C
      DATA DUPPER/0.1D60/
      DATA NPRINT/6/
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< PROGRAM BEGINS >>>>>
C
C
C-------------------------------------DEFINE SOME CONSTANTS
C
      DSTART=1.0D-10
      DK2=DK*DK
      DE=(DKAP-DK)*(DKAP+DK)
C
C-------------------------------------SET UP RUN PARAMETERS
C
    1 DUMAX=DSTART
      DMARK=1.0D0
      MS=0
C
C-------------------------------------INITIAL VALUES
C
      DUJ=DSTART
      DUPJ=DRORB*DSQRT((DK-DKMIN)*(DK+DKMIN))*DSTART
      DFJ=DE-DVZ(1)
      DUZ(1)=DUJ
C
C-------------------------------------DEPTH LOOP BEGINS
C
C
      DO 8 J=2,NP1
C
C-------------------------------------ITERATE THE FUNCTION
C
      DFJP1=DE-DVZ(J)
      DENOM=1.0D0+DH2SIX*DFJP1
      DUJP1=((1.0D0-DH2D3*DFJ)*DUJ+DH*DUPJ)/DENOM
      DA=(1.0D0-DH2D3*DFJP1)*DUPJ
      DB=(DFJ+DFJP1-DH2SIX*DFJ*DFJP1)*DUJ
      DUPJP1=(DA-DB*DHHLF)/DENOM
C
C-------------------------------------CHECK FOR MAX DU(J)
C
```

```
      DAUJP1=DABS(DUJP1)
      IF (DAUJP1-DUMAX) 4,4,2
    2 IF (DAUJP1-DUPPER) 3,3,9
    3 DUMAX=DAUJP1
      GO TO 7
C
C-----------------------------CHECK ZERO CROSSINGS
C
    4 IF (DUJP1*DMARK) 5,6,7
    5 MS=MS+1
      DMARK=DSIGN(DMARK,DUJP1)
      GO TO 7
C
    6 IF (DUJP1) 7,5,7
C
C-----------------------------SET FOR NEXT ITERATION
C
    7 DFJ=DFJP1
      DUJ=DUJP1
      DUFJ=DUPJP1
      DUZ(J)=DUJ
    8 CONTINUE
C
C
C-----------------------------DEPTH LOOP ENDS
C
C
      RETURN
C
C
C-----------------------------OVERFLOW PROTECTION
C                                  REDUCE SCALE OF U(Z)
    9 IF (DSTART.LT.1.0D-60) GO TO 10
      DSTART=DSTART*1.0D-10
      GO TO 1
C
C
   10 WRITE (NPRINT,11) DK
      MS=0
      RETURN
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< FORMAT STATEMENTS >>
C
   11 FORMAT ('1 SUBROUTINE ITERAT OVERFLOW PROTECTION',
     * ' EXCEEDED. ', //,
     1    '  MS SET TO ZERO AND RETURNED',
     * ' TO CALLING PROGRAM.', //,
     2         '  DK = ', G20.12 )
      END
```

```
      SUBROUTINE FIXUP (DKN)
      IMPLICIT REAL*8(D)
C
CCC   ***********************************************************
CCC   *
CCC   *       THIS SUBROUTINE FORCES A DECAYING EXPONENTIAL
CCC   *    ON THE EIGENFUNCTION ABOVE THE UPPER TURNING
CCC   *     POINT.
CCC   *
CCC   ***********************************************************
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< COMMON BLOCKS >>>>>>
C
      COMMON /PARAM/ N,NP1,IEX
      COMMON /DBL1/ DVZ(501),DH,DH2,DCBSQ,DRORB,DKAP,DUMAX
      COMMON /DBLE3/ DUZ(501),DZZ(501)
      COMMON /DHHH/ DH2SIX,DH2D3,DHHLF
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< PROGRAM BEGINS >>>>>
C
C
      DE=(DKAP-DKN)*(DKAP+DKN)
      NP2=NP1+1
      DARG=DE-DVZ(NP1)
      IF (DARG.GT.0.0D0) RETURN
      IPASS=-1
      IF (DABS(DUZ(NP1)).LT.DUMAX) IPASS=1
C
C-----------------------------FIND THE UPPER TURNING POINT
C
C
      DO 1 I=2,NP1
      INDEX=NP2-I
      DKZ2=DE-DVZ(INDEX)
      IF (DKZ2) 1,2,2
    1 CONTINUE
C
      RETURN
C
C     ---------------------------------DECAY THE SOLUTION
C
    2 JTPU=INDEX
      JTPUP1=JTPU+1
      ISTOP=NP1-JTPU
      DUJ=0.0D0
      DUPJ=1.0D-5
      DFJ=DE-DVZ(NP1)
C
      DO 3 J=1,ISTOP
      INDEX=NP2-J
      DUZ(INDEX)=DUJ
      DFJP1=DE-DVZ(INDEX-1)
      DENOM=1.0D0+DH2SIX*DFJP1
      DUJP1=((1.0D0-DH2D3*DFJ)*DUJ+DH*DUPJ)/DENOM
      DA=(1.0D0-DH2D3*DFJP1)*DUPJ
      DB=(DFJ+DFJP1-DH2SIX*DFJ*DFJP1)*DUJ
      DUPJP1=(DA-DB*DHHLF)/DENOM
      DFJ=DFJP1
      DUJ=DUJP1
    3 DUPJ=DUPJP1
C
      DCOEF=DUZ(JTPU)/DUJP1
C
      DO 4 I=JTPUP1,N
    4 DUZ(I)=DUZ(I)*DCOEF
C
      IF (IPASS.GT.0) RETURN
      DUMAX=0.0D0
C
      DO 6 I=1,JTPU
```

```
      DADUZI=DABS(DUZ(I))
      IF (DADUZI-DUMAX) 6,6,5
    5 DUMAX=DADUZI
    6 CONTINUE
C
CC
C
      RETURN
C
      END
```

```
      SUBROUTINE DNORM1 (DUZ,DUMAX,NP1)
      IMPLICIT REAL*8(D)
C
CCC   *
CCC   *    THIS SUBROUTINE NORMALIZES THE  DOUBLE
CCC   *    PRECISION VECTOR DUZ (NP1), TO THE VALUE
CCC   *    DUMAX.
CCC   *
C     ***********************************************************
      DIMENSION DUZ(NP1)
C
CC
      DTEST=DUMAX*0.1D-60
C
CC
      DO 1 I=1,NP1
      DUZI=DUZ(I)
      IF (DABS(DUZI).LT.DTEST) DUZI=0.0D0
    1 DUZ(I)=DUZI/DUMAX
C
CC
C
      RETURN
      END
```

```
      SUBROUTINE BOTTOM (DK,M)
      IMPLICIT REAL*8(D)
C     ***********************************************************
CC    *
CC    *        THIS SUBROUTINE COMPUTES A BOTTOM SOUND
CC    *    PROFILE FOR PLOTTING PURPOSES.
CC    *
CC    ***********************************************************
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< COMMON BLOCKS >>>>>>
C
      COMMON /DBL1/ DVZ(501),DH,DH2,DCBSQ,DRORB,DKAP,DUMAX
      COMMON /DBLE2/ DW2,DCB,DCMIN,DW,DC1,DC1SQ
      COMMON /DBLE3/ DUZ(501),DZZ(501)
      COMMON /SING1/ ZM,CB,CMIN,CMAX,PLTMAX,UM,FC
      COMMON /BOTTO/ DECAY,ZB(20),UB(20)
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< PROGRAM BEGINS >>>>>
C
C-----------------------------COMPUTE DEPTH VALUES
CC
C
      HB=(ZM-PLTMAX)*0.5E-1
      DHB=DBLE(HB)
C
C-----------------------------COMPUTE SOME CONSTANTS
C
      DGAM=DSQRT((DK-DW/DCB)*(DK+DW/DCB))
      DARG=DGAM*DHB
      IF (DEXPU+DARG) 3,3,1
    1 DEX=DEXP(DARG)
      DV=DUZ(1)*DRORB
C
C-----------------------------COMPUTE THE PROFILE
CC
C
      DO 2 I=1,20
      IF (DV.LT.1.0D0-30) GO TO 4
      DV=DV*DEX
    2 UB(I)=SNGL(DV)
C
      RETURN
C
C-----------------------------NULL PROFILE
C
    3 I=1
C
    4 DO 5 J=I,20
    5 UB(J)=0.0
C
      RETURN
      END
```

```
      SUBROUTINE INTEGR (DK,MODE)
      IMPLICIT REAL*8(D)
C     ***********************************************************
C     *
C     *  THIS SUBROUTINE COMPUTES THE NORMALIZATION
C     *  INTEGRAL, AND THE SOUND VELOCITY NORMAL-
C     *  IZATION INTEGRAL.  THEN THE GROUP AND PHASE
C     *  VELOCITIES ARE COMPUTED.
C     *
C     ***********************************************************
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< COMMON BLOCKS >>>>>>
C
      COMMON /PARAM/ N,NP1,IEX
      COMMON /SING1/ ZM,CB,CMIN,CMAX,PLTMAX,UM,FC
      COMMON /DBL1/ DVZ(501),DH,DH2,DCBSQ,DRORB,DKAP,DUMAX
      COMMON /DBLE2/ DW2,DCB,DCMIN,DW,DC1,DC1SQ
      COMMON /DBLE3/ DUZ(501),DZZ(501)
      COMMON /DBLE4/ DS,DSSCC,DPVEL,DGVEL
      COMMON /RHO/ RO,RB
      COMMON /DSOUND/ DCC(501)
      COMMON /BOTTO/ DECAY,ZB(20),UB(20)
      COMMON /DKMAP/ DKMIN,DKMAX,DKL,DKU
      EQUIVALENCE (DUZ1,DUZ(1))
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< DATA >>>>>>>>>>>>>
C
      DATA NREAD,NPRINT/5,6/
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< PROGRAM BEGINS >>>>>
C
C
C     ***   SET UP THE END POINTS FOR THE TRAPAZOIDAL RULE
C
      DUZ1SQ=DUZ(1)
      IF (DABS(DUZ1SQ).LT.1.0D-30) DUZ1SQ=0.0D0
      DUZ1SQ=DUZ1SQ*DUZ1SQ
      DS=0.5D0*(DUZ1SQ)
      DC1SQ=DCC(1)*DCC(1)
      DUZ1SQ=DUZ1SQ/(DC1SQ)
      DSSCC=0.5D0*(DUZ1SQ)
C
C-----------------------------INTEGRATING LOOP BEGINS
C
      DO 1 I=2,N
      DUZI2=DUZ(I)
      IF (DABS(DUZI2).LT.1.0D-30) GO TO 1
      DUZI2=DUZI2*DUZI2
      DS=DS+DUZI2
      DCCI=DCC(I)
      DUZI2=DUZI2/(DCCI*DCCI)
      DSSCC=DSSCC+DUZI2
    1 CONTINUE
C
C-----------------------------INTEGRATING LOOP ENDS
C
C     ***   ADD BOTTOM EXPONENTIAL DECAY   ***
C
      DA2=(DK-DKMIN)*(DK+DKMIN)
      DB=DBLE(RB)*DUZ1*0.5DC/DSQRT(DA2)
      DS=DBLE(RO)*DH*DS+DB
      DSSCC=DSSCC*DBLE(RO)*DH+DB/DCBSQ
      DECAY=DH*DB/(DCB*DK*DS)
C
C     ***  COMPUTE THE PHASE AND GROUP VELOCITIES  ***
C
```

```
      DPVEL=DW/DK
      DGVEL=DS/(DPVEL*DSSCC)
C
      WRITE (NPRINT,2) MODE,DPVEL
      WRITE (NPRINT,3) MODE,DGVEL
C
C
      RETURN
C
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< FORMAT STATEMENTS >>
C
C
    2 FORMAT ('0PHASE VELOCITY FOR MODE', I4, ' IS ',
     1        G15.7, ' M/SEC.')
    3 FORMAT ('0GROUP VELOCITY FOR MODE', I4, ' IS ',
     1        G15.7, ' M/SEC.')
      END
```

```
      SUBROUTINE MODPLT (M,JUMP)
      IMPLICIT REAL*8(D)
C
CC    ***********************************************************
CC    *
CC    *   THIS SUBROUTINE PLOTS THE MODE SHAPE
CC    *   ON THE PRINTER USING THE N.P.S. ROUTINE
CC    *   UTPLOT .
CC    *
CC    *        IF JUMP=1, ALL POINTS IN THE MODE
CC    *        FUNCTION WILL BE PLOTTED.  IF JUMP
CC    *        IS NOT EQUAL TO 1, EVERY NP1/100 TH
CC    *        POINT WILL BE PLOTTED.
CC    *
CC    ***********************************************************
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< COMMON BLOCKS >>>>>>>
C
      COMMON /DBL1/ DVZ(501),DH,DH2,DCBSQ,DRORB,DKAP2,DUMAX
      COMMON /DBLE2/ DW2,DCB,DCMIN,DW,DC1,DC1SQ
      COMMON /DBLE3/ DUZ(501),DZZ(501)
      COMMON /PARAM/ N,NP1,IEX
      COMMON /HEAD/ HED(20)
      COMMON /SING1/ ZM,CB,CMIN,CMAX,PLTMAX,UM,FC
      COMMON /DMAP/ DK1(100)
      COMMON /PLOT6/ RANGE(4)
      COMMON /INPUT/ NUMV,NMOD
      COMMON /BOTTO/ DECAY,ZB(20),UB(20)
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< PROGRAM BEGINS >>>>>>
C
C
      DK=DK1(M)
C
CC    ***  CHECK IF FULL PLOT OF ALL POINTS DESIRED  ***
C
      IF (JUMP-1) 1,2,1
    1 JUMP=NP1/100
C
    2 JUMP2=JUMP*2
      NUMB=NP1/JUMP
C
CC    ***  PRINT THE HEADINGS  **
C
      WRITE (6,6) HED,FC
      WRITE (6,7) M,DK1(M)
      WRITE (6,8)
C
CC    ***  COMPUTE THE RANGE FACTORS FOR PLOT SIZE  ***
C
      RANGE(1)=1.1
      RANGE(2)=-1.1
      RANGE(3)=ZM
C
CC    ***  WILL BOTTOM PROFILE BE INCLUDED?  **
C
      IF (ZM-PLTMAX) 4,3,3
C
    3 MODCUR=0
      RANGE(4)=0.0
      GO TO 5
C
CC    ***  DRAW BOTTOM PROFILE  ***
C
    4 MODCUR=1
      RANGE(4)=ZM-PLTMAX
C
      CALL UTPLOT (UB,ZB,20,RANGE,1,MODCUR)
      MODCUR=3
C
CC
C     ***  DRAW OCEAN U(Z) PROFILE  ***
```

```
C
      5 CALL UTPLOT (DUZ,DZZ,NUMB,RANGE,JUMP2,MODCLR)
C
C
        RETURN
C
      6 FORMAT ('1', 20A4, ' FREQUENCY : ', F6.1, ' HZ')
      7 FORMAT ('0', T25, 'MODE', I4, '   :  K =', F16.12)
      8 FORMAT ('0', //, T23, 'NORMALIZED EIGENFUNCTION ',
     1       ' VS DEPTH')
        END
```

```
      SUBROUTINE OUTPUT (DUZ,DK,MODE,NPTS)
      IMPLICIT REAL*8(D)
      DIMENSION DZZ(NPTS), DUZ(NPTS), DCC(NPTS)
      COMMON /RHO/ RO,RB
      COMMON /SING1/ ZM,CB,CMIN,CMAX,PLTMAX,UM,FQ
      COMMON /BOTTO/ DECAY,ZB(20),UB(20)
      COMMON /DBLE4/ DS,DSSQC,DPVEL,DGVEL
      COMMON /HEAD/ HED(20)
      COMMON /WORK/ EXTRA(20)
      DATA NPRINT,NPUNCH/6,7/
C
C
    1 WRITE (NPUNCH,10) MODE,FQ,DK,DPVEL,DGVEL,DS,DECAY
C
      DO 2 I=1,20
    2 EXTRA(I)=UB(21-I)
C
      WRITE (NPUNCH,11) EXTRA
      WRITE (NPUNCH,11) DUZ
      WRITE (NPRINT,13) MODE
      RETURN
C
C
      ENTRY PROFIL(DZZ,DCC,NPTS)
      WRITE (NPUNCH,7) HED
      NPTS20=NPTS+20
      WRITE (NPUNCH,8) NPTS20,RO,RB,CMIN,CB,ZM,PLTMAX
C
      DO 3 I=1,20
    3 EXTRA(I)=ZM-ZB(21-I)
C
      WRITE (NPUNCH,12) EXTRA
      DZM=DBLE(ZM)
C
      DO 4 I=1,NPTS
    4 DZZ(I)=DZM-DZZ(I)
C
      WRITE (NPUNCH,12) DZZ
C
      DO 5 I=1,20
    5 EXTRA(I)=CB
C
      WRITE (NPUNCH,12) EXTRA
      WRITE (NPUNCH,12) DCC
      WRITE (NPRINT,9)
C
      DO 6 I=1,NPTS
    6 DZZ(I)=DZM-DZZ(I)
C
      RETURN
C
    7 FORMAT (20A4)
    8 FORMAT (I10,2F10.5,4F10.2)
    9 FORMAT ('OSCUND VELOCITY DETAILED PROFILE PUNCHED')
      END
```

```
      SUBROUTINE CZPLOT
      IMPLICIT REAL*8(D)
C
CC    ********************************* **********************
CC    *    THIS SUBROUTINE PLOTS THE SOUND VELOCITY
CC    *    PROFILE ON THE PRINTER PLOT USING
CC    *    THE N.P.S. ROUTINE UTPLOT.
CC    *
CC    ***************************************************
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< COMMON BLOCKS >>>>>
C
      COMMON /DBL1/ DVZ(501),DH,DH2,DCBSQ,DRORB,CKAP2,DUMAX
      COMMON /DBLE2/ DW2,DCB,DCMIN,DW,DC1,DC1SQ
      COMMON /DBLE3/ DUZ(501),DZZ(501)
      COMMON /PARAM/ N,NP1,IEX
      COMMON /HEAD/ HED(20)
      COMMON /WORK/ WRK1(20)
      COMMON /SING1/ ZM,CB,CMIN,CMAX,PLTMAX,UM,FQ
      COMMON /PLOT6/ RANGE(4)
      COMMON /DSOUND/ DCC(501)
      COMMON /BOTTO/ DECAY,ZB(20),UB(20)
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< PROGRAM BEGINS >>>>>>
C
CC    ***  WRITE THE HEADINGS  ***
C
      WRITE (6,5) HED
      WRITE (6,6)
      WRITE (6,7)
C
      IF (PLTMAX.LT.ZM) PLTMAX=ZM
      FE=(ZM-PLTMAX)*0.5D-1
C
CC    ------------------------ DEPTH INCREMENT LOOP BEGINS
C
      DO 1 I=1,20
      WRK1(I)=CB
      ZB(I)=HB*FLOAT(I-1)
    1 CONTINUE
C
CC    ---------------------- DEPTH LOOP ENDS
CC
CC    ***  COMPUTE THE RANGE FACTORS  ***
C
      RANGE(1)=AMAX1(CB,CMAX)
      RANGE(2)=CMIN
      RANGE(3)=ZM
C
CC    ***  WILL BOTTOM PROFILE BE DRAWN?  ***
C
      IF (ZM-PLTMAX) 3,2,2
C
C
    2 MODCUR=0
      RANGE(4)=0.0
      GO TO 4
C
    3 MODCUR=1
      RANGE(4)=ZM-PLTMAX
C
CC    ***  DRAW THE BOTTOM C(Z) PROFILE ***
C
      CALL UTPLOT (WRK1,ZB,20,RANGE,1,MODCUR)
      MODCUR=3
C
CC    ***  DRAW THE OCEAN C(Z) PROFILE ***
CC
C
    4 CALL UTPLOT (DCC,DZZ,NP1,RANGE,2,MODCUR)
C
      RETURN
```

```
C
    5 FORMAT ('1', 20A4)
    6 FORMAT ('0', //, T28, 'PLOT OF C(Z)  VS  DEPTH')
    7 FORMAT ('0', //, T11, 'BOTTOM AT ZERO. DEPTH IN ',
     1 'METERS ABOVE BOTTOM.  C(Z) IN METERS/SEC')
      END
```

```
      IMPLICIT REAL*8 (A-H, V-Z), LOGICAL*4 ($)
C
C
C
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<COMMON BLOCKS>>>>>>>>>>>>>>>>
C
      COMMON / D1 / VELC(1021), DEPTH (1021)
      COMMON / D2 / ZZZ(1021), DVZ(1001)
      COMMON / D3 / CC(50),DEP(50)
      COMMON / D4 / DRO,DRB,DRORB
      COMMON / D5 / CMAX,CMIN,CB,CBSQ
      COMMON / D6 / H,DH,HPLOT,DHHLF,DH2SIX,DH2C3,
     1              DH2,HBOTT,DHB
      COMMON / D7 / DK,DKMIN,DKMAX,DKOLD,DKU,DKL,
     2              DTEST,DSMAX,DE
      COMMON / D9 / ZMAX,DSS,DECAY,DPVEL,DGVEL,GAM
      COMMON / D10/ WOCB,WCCNP1,WOCMIN,DVZB
      COMMON / DW / FQV(20),FQ,W,W2
      COMMON /AWKB/ DKN,DINT,DIFF,DS,DM
      COMMON/DKMAP/ DK1(100)
      COMMON /HEAD/ HED(20)
      COMMON / I1 / NUMV,NMOD,NFREQ,NMODF,N,NP1,
     3              NP21,NPTS,MODE,IT,NP2
      COMMON / I2 / LOMODE,IHMODE,KWELL,IWELL,NWELL,
     4              JHI,JLOW
      COMMON / I3 / JUPPER,JBOTT,ISHAP,JTPL,JTPU
      COMMON / IO / NREAD,NPRINT,NPUNCH
      COMMON / L1 / $GRAPH,$BOTPR,$CARDS,$CEL
      COMMON / P1 / DPI,DPIC2,DPIC4,DPIRT2,D2PI
C
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< PROGRAM BEGINS >>>>>>>>>>>>>>>>
C
C
      WRITE (NPRINT,6)
      CALL INPUT2
      CALL LINEAR
      IF ($GRAPH) CALL CZPLOT
      IF ($CEL.OR.$CARDS) CALL PROFIL (DEPTH,VELC,NPTS)
C
C-----------------------------FREQUENCY LOOP BEGINS
C
      DO 5  IFREQ=1,NFREQ
C
      FQ=FQV(IFREQ)
      W=FQ*D2PI
      W2=W*W
      WCCB=W/CB
      WCCMIN=W/CMIN
C
C-----------------------------COMPUTE VELOCITY FUNCTION
C
      DO 1 J=1,NP1
      WCC=W/VELC(J)
    1 DVZ(J)=(WCCMIN-WCC)*(WCC+WOCMIN)
C
      WCCNP1=WCC
      DVZB=(WCCMIN-WOCB)*(WCCMIN+WOCB)
      CALL MAXMI3 (65)
```

```
      KWELL=1
      LCMODE=0
C
C-----------------------------MODE LOOP BEGINS
C
      DO 4 MODE=1,NMODE
      CALL SEARCH
      CALL SHAPE4
      CALL INTEGR
      IF ($GRAPH) GO TO 2
      IF (MODE.EQ.1) WRITE (NPRINT,7) HED
      WRITE (NPRINT,8) MODE,DK,DPVEL,DGVEL,DSS,DECAY
      GO TO 3
C
    2 CALL MODPLT
    3 IF ($CEL.OR.$CARDS) CALL OUTPUT (ZZZ,NPTS)
C
    4 CONTINUE
C
    5 CONTINUE
C
C
C-----------------------------FREQUENCY AND MODE LOOPS END
C
      STOP
C
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< FORMAT STATEMENTS >>
C
    6 FORMAT ('1 NORM04 : WKB METHOD NORMAL MODE',
     1  ' EIGENVALUE SOLVER. ')
    7 FORMAT ('1',10A8,//,'  MODE',T25,'HORIZ WAVE NO.',T40,
     1'PHASE SPEED',T60,'GROUP SPEED',T84,'DSS',T103,
    8 FORMAT ('0',I4,F20.12,' 1/METERS',T40,F7.1,' M/SEC,
     1T60,F7.1,' M/SEC',T76,2G20.12)
      END
```

```
      BLOCK DATA
      IMPLICIT REAL*8 (A-H, V-Z), LOGICAL*4 ($)
      COMMON / D5 / CMAX,CMIN,CB,CBSQ
      COMMON /AWK6/ DKN,DINT,DIFF,DS,DM
      COMMON / P1 / DPI,DPIC2,DPIC4,DPIRT2,D2PI
      COMMON / I1 / NUMV,NMOD,NFREQ,NMCDF,N,NP1,
     3              NP21,NPTS,MODE,IT,NP2
      COMMON / IO / NREAD,NPRINT,NPUNCH
      COMMON / L1 / $GRAPH,$BCTPR,$CARDS,$CEL
      DATA NREAD, NPRINT, NPUNCH /5, 6, 7/
      DATA $GRAPH,$CARDS,$CEL,$BCTPR /3*.FALSE.,.TRUE./
      DATA CMAX,CMIN /0.0D0, 9.9D9/
      DATA MODE /0/, DM /0.0D0/
      DATA DPIO2 /1.5707963267948966/
      DATA D2PI  /6.2831853C7179586/
      DATA DPIO4 /0.7853981633974483/
      DATA DPI /3.141592653589793/
      END
```

```
      SUBROUTINE MAXMI3 (*)
      IMPLICIT REAL*8 (A-H, V-Z), LOGICAL*4 ($)
C
C     -----------------------------------
C     LAST CHANGED 3 AUG 73
C     -----------------------------------
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< COMMON BLOCKS >>>>>>
C
      COMMON / D5 / CMAX,CMIN,CB,CBSQ
      COMMON / D7 / DK,DKMIN,DKMAX,DKCLD,DKU,DKL,
     2              DTEST,DSMAX,DE
      COMMON / DW / FQV(20),FQ,W,W2
      COMMON /AWKB/ DKN,DINT,DIFF,DS,DM
      COMMON /HEAD/ HED(20)
      COMMON / I1 / NUMV,NMOD,NFREQ,NMODF,N,NP1,
     3              NP21,NPTS,MODE,IT,NP2
      COMMON / IO / NREAD,NPRINT,NPUNCH
      COMMON / L1 / $GRAPH,$BCTPR,$CARDS,$CEL
      COMMON / P1 / DPI,DPI02,DPI04,DPIRT2,D2PI
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< PROGRAM BEGINS >>>>>
C
      DM=0.0D0
      DKMIN=W/CB
      DKMAX=W/CMIN
      DKN=DKMIN
      CALL WKB
      DSMAX=DS/DPI
      MODMAX=IDINT(DSMAX)
      DKCLD=DKMAX
      WRITE (NPRINT,3) HED,FQ
      DKN=DKMAX
      MODMIN=0
      WRITE (NPRINT,4)
      DATA MAXI/'MAXI'/, MINI/'MINI'/
      WRITE (NPRINT,5) MAXI,DKMAX,MODMIN
      WRITE (NPRINT,5) MINI,DKMIN,MODMAX
C
C------------------------------CHECK NUMBER MODES REQUESTED
C
      NMODF=NMOD
      IF (MODMAX-NMOD) 1,2,2
C
    1 CONTINUE
C
C------------------------------IF WE GET HERE WE MUST RESET
C                              THE NUMBER OF MODES REQUESTED
C
      NMODF=MODMAX
C
      WRITE (NPRINT,6)
      WRITE (NPRINT,7)
      WRITE (NPRINT,8)
      WRITE (NPRINT,7)
      WRITE (NPRINT,6)
      WRITE (NPRINT,9) NMODF
C
    2 IF (MODMAX.NE.0) RETURN
C
C------------------------------IF WE GET HERE WE ARE BELOW
C                              CUT - OFF FREQUENCY
C
      WRITE (NPRINT,6)
      WRITE (NPRINT,7)
      WRITE (NPRINT,8)
      WRITE (NPRINT,7)
      WRITE (NPRINT,6)
      WRITE (NPRINT,10)
C
      RETURN 1
C
```

```
C
C
    3 FORMAT ('1', 10A8, 4X, 'FREQUENCY : ', F6.1)
    4 FORMAT('0',//,T25,'BOUNDS ON HORIZONTAL WAVE NUMBER')
    5 FORMAT ('0', T22, A4, 'NUM K :', F12.8,3X,'MODES :',I4
    6 FORMAT (T33,13('*'))
    7 FORMAT (T33,1H*,11X,1H*)
    8 FORMAT (T33,'*  CAUTION  *')
    9 FORMAT ('0', T26, 'MODES REQUESTED RESET TO :', I4)
   10 FORMAT ('0',/,T25, 'FREQUENCY REQUESTED IS BELOW',
     1   ' CUT-OFF',//,T20,'JOB TERMINATED OR NEW FREQ',
     2   'UENCY SELECTED')
      END
```

```
      SUBROUTINE INTEGR
      IMPLICIT REAL*8 (A-H, V-Z), LOGICAL*4 ($)
C     ---------------------------------
CC    LAST CHANGED 9 AUG 73
C     ---------------------------------
      COMMON / D1 / VELO(1021), DEPTH (1021)
      COMMON / D2 / ZZZ(1021), DVZ(1001)
      COMMON / D4 / DRO,DRB,DRORB
      COMMON / D5 / CMAX,CMIN,CB,CBSQ
      COMMON / D6 / H,DH,HPLCT,DHHLF,DH2SIX,DH2D3,
     1              DH2,HBOTT,DHB
      COMMON / D7 / DK,DKMIN,DKMAX,DKOLD,DKU,DKL,
     2              DTEST,DSMAX,DE
      COMMON / D9 / ZMAX,DSS,DECAY,DPVEL,DGVEL,GAM
      COMMON / D10/ WOCB,WCCNP1,WCCMIN,DVZB
      COMMON / DW / FQV(20),FQ,W,W2
      COMMON / I1 / NUMV,NMOD,NFREQ,NMCDF,N,NP1,
     3              NP21,NPTS,MODE,IT,NP2
C
C
      DSS=0.0D0
      DSSCC=0.0D0
C
CC
CC    <<<<<<<<<<<<<<<<<<<<<<<<<<<<<< INTEGRATING LOOP BEGINS
CC
C
      DO 1 I=2,N
C
      ZZZI2=ZZZ(I)
      IF (DABS(ZZZI2).LT.1.0D-30) GO TO 1
      ZZZI2=ZZZI2*ZZZI2
C
      DSS=DSS+ZZZI2
C
      VELOI=VELO(I)
      ZZZI2=ZZZI2/(VELOI*VELOI)
C
      DSSCC=DSSCC+ZZZI2
C
    1 CONTINUE
C
CC    <<<<<<<<<<<<<<<<<<<<<<<<<<<<<< INTEGRATING LOOP ENDS
CC
C
      IF (DABS(ZZZ(NP1)).LT.1.0D-30) GO TO 3
      A2=DSQRT(DVZB-DE)
      B=ZZZ(NP1)*ZZZ(NP1)*DRORB*DRO/(2.0D0*A2)
    2 DSS=DRO*DH*DSS+B
      DSSCC=DSSCC*DRO*DH+B/CBSQ
      DECAY=W*B/(CB*DK*DSS)
C
      DPVEL=W/DK
      DGVEL=DSS/(DPVEL*DSSCC)
      RETURN
C
    3 B=0.0D0
      GO TO 2
C
      END
```

```
      SUBROUTINE WKB
      IMPLICIT REAL*8 (A-H,V-Z),LOGICAL*4 ($)
C
C     ---------------------------------
      LAST CHANGED : 7 SEPT 73  ASYMPTOTIC B. C.
C     ---------------------------------
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< COMMON BLOCKS >>>>>>
C
      COMMON / D2 / ZZZ(1021), DVZ(1001)
      COMMON / D4 / DRO,DRB,DRORB
      COMMON / D6 / H,DH,HPLOT,DHHLF,DH2SIX,DH2D3,
     1              DH2,HBOTT,DHB
      COMMON / D7 / DK,DKMIN,DKMAX,DKOLD,DKU,DKL,
     2              DTEST,DSMAX,DE
      COMMON / D10/ WOCB,WOCNP1,WOCMIN,DVZB
      COMMON /AWKB/ DKN,DINT,DIFF,DS,DM
      COMMON / I1 / NUMV,NMOD,NFREQ,NMODF,N,NP1,
     3              NP21,NPTS,MODE,IT,NP2
      COMMON / I2 / LOMODE,IHMODE,KWELL,IWELL,NWELL,
     4              JHI,JLOW
      COMMON / I3 / JUPPER,JBOTT,ISHAP,JTPL,JTFL
      COMMON / L1 / $GRAPH,$BOTPR,$CARDS,$CEL
      COMMON / P1 / DPI,DPIC2,DPIC4,DPIRT2,D2PI
      COMMON / DTH/ DTHETL
      DATA DEXPU /1.7D2/, DWELL /6.0/
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< PROGRAM BEGINS >>>>>
C
      DINT=0.0D0
      JTPL=1
      DE=(DKMAX-DKN)*(DKMAX+DKN)
      DKB=DSQRT(DVZB-DE)
      $TOP=.FALSE.
      IWELL=1
      ISIG=-1
      DARG=0.0D0
      DARGP=0.0D0
C
C     ---------------------------------DEPTH LOOP BEGINS
C
      DO 21 J=1,NP1
      DKZ2=DE-DVZ(J)
C
C     ---------------------------------CHECK FOR CONDITIONS
C                                       AND TYPE OF SOLUTION
C
      IF (DKZ2) 1,2,3
    1 IF (ISIG) 4,5,6
    2 IF (ISIG) 7,21,9
    3 IF (ISIG) 10,19,20
C
C     ---------------------------------IMAGINARY REGION
C
    4 DARGP=DSQRT(-DKZ2)
      DARG=DARG+DARGP
      GO TO 21
C
    5 ISIG=-1
      GO TO 4
C
    6 DARGP=DSQRT(-DKZ2)
      DX=DKZ/(DKZ+DARGP)
      DINT=DINT+DKZ*(DX-1.0D0)*0.5D0*DH
      ISIG=-1
      JTPL=J-1
      DARG=(2.0D0-DX)*DARGP*0.5D0
      GO TO 21
C
C     ---------------------- ZERO VERTICAL WAVE NUMBER
C
```

```
    7 IBACK=-1
    8 ISIG=0
      GO TO 21
C
    9 IBACK=1
      GO TO 8
C
CC    ----------------------------------REAL REGION
C
   10 ISIG=1
      IF ($TOP) GO TO 11
      JTPO=J
   11 DKZ=DSQRT(DKZ2)
      DX=DARGP/(DKZ+DARGP)
      DARG=DARG+DARGP*(DX-1.0D0)*0.5D0
      IF ($TOP) GO TO 13
   12 DTHETU=DATAN(DTANH(DARG*DH))
      DINT=DTHETU
      $TOP=.TRUE.
      GO TO 18
   13 DT1=DMOD(DINT,D2PI)
      IF (DARG*DH-DWELL) 15,14,14
   14 IF (IWELL.EQ.KWELL) GO TO 28
      DARG=0.0D0
      IWELL=IWELL+1
      JTPO=J
      DTHETU=DPI04
      DINT=DTHETU
      GO TO 18
   15 A=DSIN(DT1)
      B=DCOS(DT1)
      DEL=DEXP(DARG*DH)
      DEML=DEXP(-DARG*DH)
      AC=DEL*(A+B)+DEML*(A-B)
      BC=DEL*(A+B)-DEML*(A-B)
      DINTP=DINT-DT1+DATAN2(AC,BC)
   16 IF (DINTP.GE.DINT-DTEST) GO TO 17
      DINTP=DINTP+DPI02
      GO TO 16
   17 DINT=DINTP
   18 DINT=DINT+(2.0D0-DX)*DKZ*0.5D0*DH
      DARG=0.0D0
      GO TO 21
C
   19 DKZ=DSQRT(DKZ2)
      ISIG=+1
      DINT=DINT+DKZ*DH
      IF ($TOP) GO TO 13
      JTPO=J
      GO TO 12
C
   20 DKZ=DSQRT(DKZ2)
      DINT=DINT+DKZ*DH
C
   21 CONTINUE
C
CC
CC    ----------------------------------DEPTH LOOP ENDS
C
      IF (DKZ2) 22,27,25
C
CC    ----------------------------------DECAYING SOLUTION
C
   22 DKS=DARGP
      DARG=(DARG-0.5D0*DARGP)*DH*2.0D0
      IF (DARG-DEXPO) 23,24,24
   23 D2=(DKS-DRORB*DKB)*DEXP(-DARG)/(DKS+DRORE*DKB)
      DTHETL=DATAN((1.0D0+D2)/(1.0D0-D2))
      GO TO 27
   24 DTHETL=DPI04
      GO TO 27
C
```

```
C      ------------------------------BOTTOM REFLECTED
C
   25 JTPL=NP1
      IF (DKB.EQ.0.0D0) GO TO 26
      DTHETL=DATAN(DKZ/(DKB*DRORB))
      GO TO 27
C
   26 DTHETL=DPIC2
C
C      ------------------------------COMPUTE DIFFERENCE
C
   27 DS=DINT+DTHETL
      DINT=DINT-DTHETU
      DIFF=DS-DM*DPI
      NWELL=IWELL
      RETURN
C
   28 IWELL=IWELL+1
      GO TO 24
C
      END
```

```
      SUBROUTINE SEARCH
      IMPLICIT REAL *8 (A-H, V-Z), LOGICAL*4 ($)
C
CCC   ---------------------------------
CCC   LAST CHANGE : 15 AUGUST 1973
CCC   ---------------------------------
CCC
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< COMMON BLOCKS >>>>>>
C
      COMMON / D7 / DK,DKMIN,DKMAX,DKOLD,DKU,DKL,
     2              DTEST,DSMAX,DE
      COMMON / D8 / DIFFI
      COMMON/DKMAP/ DKL(100)
      COMMON /AWKB/ DKN,DINT,DIFF,DS,DM
      COMMON / I1 / NUMV,NMOD,NFREG,NMODF,N,NP1,
     3              NP21,NPTS,MODE,IT,NP2
      COMMON / I2 / LOMODE,IHMODE,KWELL,IWELL,NWELL,
     4              JHI,JLOW
      COMMON / I3 / JUPPER,JBOTT,ISHAP,JTPL,JTPU
      COMMON / IO / NREAD,NPRINT,NPUNCH
      COMMON / L1 / $GRAPH,$BCTPR,$CARDS,$CEL
      COMMON / P1 / DPI,DPI02,DPI04,DPIRT2,D2PI
      COMMON / DTH/ DTHETU
C
      DATA ITMAX /30/
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< PROGRAM BEGINS >>>>>
CCC
CCC   ---------------------------------INITIAL SETTINGS
C
      FIX=1.4D0
      IF (MODE.GT.1) GO TO 1
      KWELL=1
      $LOW=.FALSE.
      $HI=.FALSE.
      DKOLD=DKMAX
      JUPPER=1
      JHI=1
      JLOW=NP1
      IHMODE=0
      LOMODE=0
      DKCEN=DKOLD
      DKCEN2=DKCEN
      JBOTT=NP1
C
CCC   ---------------------------------SET UP FOR FIRST GUESS
C
    1 DM=DFLOAT(MODE-LOMODE-IHMODE)
      IT=0
      IF ($LOW) GO TO 11
      DKGUES=(DKOLD-DKMIN)/(DSMAX-DM+1.0D0)
    2 DKL=DKOLD-DKGUES*FIX
      IF (DKL.LT.DKMIN) DKL=DKMIN
      DKU=DKOLD
      FL=-DPI
    3 DKN=DKL
      CALL WKB
      IF (JTPL.LT.JHI) GO TO 5
      IF (DIFF.LT.0.0D0) GO TO 4
      FL=DIFF
      JHI=JTPL
      JLOW=JTPL
      GO TO 16
C
C
    4 FIX=FIX+0.4D0
      GO TO 2
C
CCC   ---------------------------------UPPER CHANNEL SET-UP
C
```

```
     5 IF (DS-DFLOAT(IHMODE+1)*DPI) 6,32,32
     6 KWELL=2
       GO TO 3
C
    32 $HI=.TRUE.
       JUPPER=JTPL
       IHMODE=IHMODE+1
       DM=DFLOAT(IHMODE)
     7 DKN=(DKU+DKL)*0.5D0
     8 CALL WKB
       IF (JTPL.LT.JUPPER) GO TO 9
       DKU=DKN
       GO TO 7
     9 IF (DIFF.LT.0.0D0) GO TO 10
       DKN=(DKU+DKN)*0.5D0
       GO TO 8
    10 DKU=DKN
       FU=DIFF
       GO TO 16
C
CC     ------------------------------------LOWER CHANNEL SET-UP
C
    11 DKL=DK
       DM=DFLOAT(LOMODE)
       DKL=DKCEN2
       DKN=(DKU+DKL)*0.5D0
    12 CALL WKB
       IF (JTPL.GT.JBOTT) GO TO 13
       DKN=(DKL+DKN)*0.5D0
       GO TO 12
    13 FL=DIFF
       DKU=DKN
    14 IF (FL.GT.0.0D0) GO TO 16
       DKL=DKL-.25D0*(DKMAX-DKMIN)/DSMAX
       IF (DKL.LT.DKMIN) DKL=DKMIN
       DKN=DKL
       CALL WKB
       IF (JTPL.LT.JLOW) GO TO 15
       FL=DIFF
       GO TO 14
    15 KWELL=1
       $LOW=.FALSE.
       LOMODE=LOMODE-1
       GO TO 1
C
CC     ------------------------------------COMPUTE NEXT DKN
C
    16 DKN=DKL+FL*(DKU-DKL)/(FL-FU)
C
CC     ------------------------------------CALL WKB
C
    17 CALL WKB
       IT=IT+1
       IF ($LOW.OR.$HI) GO TO 18
       IF (JTPL.LT.JUPPER) DIFF=DIFF-DFLOAT(IHMODE)*DPI
       IF (JTPL.GT.JBOTT) DIFF=DIFF-DFLOAT(LOMODE)*DPI
C
CC     ------------------------CHECK WHETHER ABOVE OR BELOW
C
    18 IF (DIFF) 19,20,21
    19 DKU=DKN
       FU=DIFF
       GO TO 22
C
    20 WRITE (NPRINT,31) MODE
       GO TO 25
C
    21 DKL=DKN
       FL=DIFF
C
CC     ------------------------------------TEST THE RESULT
C
```

```
   22 DINC=DABS(DKOLD-DKN)/DKN
      DKOLD=DKN
      IF (DABS(DIFF)-DTEST) 25,25,23
   23 IF (MOD(IT,10).EQ.9) GO TO 24
      IF (IT.LE.ITMAX) GO TO 16
      IF (DINC.LT.1.0D-15) GO TO 25
   24 DKN=(DKU+DKL)*0.5D0
      GO TO 17
C
CC    ------------------------------THE MODE HAS BEEN FOUND
C
   25 DK1(MODE)=DKN
      DK=DKN
      DIFFI=DIFF
      IF ($HI) GO TO 26
      IF ($LOW) GO TO 27
C
CC    ----------------------------------CENTER WELL
C
      DKCEN2=DKCEN
      DKCEN=DKN
      JHI=JTPU
      JLOW=JTPL
      IF (JTPU.LT.JUPPER) IHMODE=0
      IF (KWELL.LT.NWELL) GO TO 28
      KWELL=1
      IF (JTPL.GT.JBOTT) LOMODE=0
      ISHAP=1
      RETURN
C
CC    ----------------------------------UPPER WELL
C
   26 DKOLD=DKCEN
      JUPPER=JHI
      $HI=.FALSE.
      KWELL=2
      ISHAP=2
      RETURN
C
CC    ----------------------------------LOWER WELL
C
   27 DKOLD=DKCEN
      KWELL=1
      $LOW=.FALSE.
      ISHAP=4
      RETURN
C
CC    ------------------------CHECK FOR MODE IN LOWER DUCT
C
   28 KWELL=KWELL+1
      ISHAP=3
      J1=JTPU
      DS1=DS
      J2=JTPL
      CALL WKB
      LODIFF=DS-DFLOAT(LOMODE+1)*DPI
      IF (LODIFF+10.0D0*DTEST) 29,30,30
   29 KWELL=1
      JTPL=J2
      DS=DS1
      JTPU=J1
      RETURN
C
CC    ------------------------SET UP FOR LOWER CHANNEL
C
   30 LOMODE=LOMODE+1
      JBOTT=JTPU
      FL=LODIFF
      $LOW=.TRUE.
      JTPU=J1
      DS=DS1
      JTPL=J2
```

```
      RETURN
C
   31 FORMAT ('OMCDE ',I4,' FOUND BY DIRECT HIT.')
      END
```

```
      SUBROUTINE SHAPE4
      IMPLICIT REAL*8 (A-H,V-Z), LOGICAL*4 ($)
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< COMMON BLOCKS >>>>>>
C
      COMMON / D2 / ZZZ(1021), DVZ(1001)
      COMMON / D4 / DRO,DRB,DRORB
      COMMON / D5 / CMAX,CMIN,CB,CBSQ
      COMMON / D6 / H,DH,HPLOT,DHHLF,DH2SIX,DH2D3,
     1              DH2,HBOTT,DHB
      COMMON / D7 / DK,DKMIN,DKMAX,DKCLC,DKU,DKL,
     2              DTEST,DSMAX,DE
      COMMON / D9 / ZMAX,CSS,DECAY,DPVEL,DGVEL,GAM
      COMMON / D10/ WOCB,WCCNP1,WOCMIN,DVZB
      COMMON / DW / FQV(20),FQ,W,W2
      COMMON /AWKB/ DKN,DINT,DIFF,DS,DM
      COMMON / I1 / NUMV,NMOD,NFREQ,NMODF,N,NP1,
     3              NP21,NPTS,MODE,IT,NP2
      COMMON / I2 / LOMODE,IHMODE,KWELL,IWELL,NWELL,
     4              JHI,JLOW
      COMMON / I3 / JUPPER,JBOTT,ISHAP,JTPL,JTPU
      COMMON / IC / NREAD,NPRINT,NPUNCH
      COMMON / L1 / $GRAPH,$BOTPR,$CARDS,$CEL
      COMMON / DTH/ DTHETU
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< DATA >>>>>>>>>>>>>>
C
      DATA DUPPER /9.99D 60/
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< PROGRAM BEGINS >>>>>
C
      DSTART=1.0D0
      DE=(DKMAX-DK)*(DKMAX+DK)
      IF (ISHAP.EQ.3.AND.IHMODE.GE.1) ISHAP=6
      IF (ISHAP.EQ.1.AND.IHMODE.GE.1) ISHAP=5
      $GC=.FALSE.
      ISTART=JTPU
      ISTOP=NP1
      GO TO (1,1,1,2,3,3), ISHAP
    1 ITOP=1
      GO TO 4
    2 ITOP=JLOW+1
      GO TO 4
    3 ITOP=JUPPER
    4 GO TO (7,5,6,7,7,6), ISHAP
    5 ISTOP=JHI-1
      GO TO 7
    6 ISTOP=JBOTT-1
    7 CONTINUE
      AVKZ=DINT/(DFLOAT(JTPL-JTPU)*DH)
      ZJST=DSIN(DTHETU)*DSTART
      FACTOR=ZJST*(DVZ(ISTART)-DVZ(ISTART-1))/(AVKZ*4.D0*DH)
      ZPJST=AVKZ*DCOS(DTHETU)*DSTART-FACTOR
      ZJ=ZJST
      ZPJ=ZPJST
C
      FJ=DE-DVZ(ISTART)
      JSTART=ISTART
      JSTOP=ISTOP
      DENOM=1.0D0-DH2SIX*(DE-DVZ(JSTART-1))
      ZJM1=((1.0D0-DH2D3*FJ)*ZJ-DH*ZPJ)/DENOM
C
C
C     ----------------------------------DEPTH LOOP BEGINS
C
      ZMAX=0.0D0
C
      DO 10 J=JSTART,JSTOP
      ZZZ(J)=ZJ
      FJ=DE-DVZ(J)
      ZJP1=(2.0D0-DH2*FJ)*ZJ-ZJM1
      ZJM1=ZJ
```

```
      AZJ=DABS(ZJ)
C
      IF (J-JTPL) 9,9,8
C
    8 IF (AZJ.LT.ZMAX*DTEST) GO TO 11
      IF (DABS(ZJP1).GT.AZJ) GO TO 11
C
    9 IF (AZJ.GT.DUPPER) GO TO 29
      IF (AZJ.GT.ZMAX) ZMAX=AZJ
   10 ZJ=ZJP1
C
C
C     ----------------------------------DEPTH LOOP ENDS
C
      IF (ISTOP.EQ.NP1) GO TO 16
      JSTART=JSTOP+1
      GO TO 12
   11 JSTART=(J+JTPL)/2
   12 ZJ=ZZZ(JSTART)
C
      DO 13 J=JSTART,JSTOP
      ZZZ(J)=ZJ
      DKZ=DSQRT(DABS(DVZ(J)-DE))
      ZJ=ZJ*DEXP(-DKZ*DH)
      IF (DABS(ZJ).LT.ZMAX*DTEST) GO TO 14
   13 CONTINUE
C
      IF (ISTOP.EQ.NP1) GO TO 16
      J=JSTOP
   14 JSTART=J+1
      $GO=.TRUE.
C
      DO 15 J=JSTART,NPTS
   15 ZZZ(J)=0.0D0
C
C
C     ---------------------COMPUTE UPPER DECAYING SOLUTION
C
   16 IF (ISTART.EQ.ITOP) GO TO 24
      ZJ=ZJST
      ZPJ=-ZPJST
      FJ=DE-DVZ(ISTART)
      ZJM1=ZJ-DH*ZPJ-DH2*FJ*0.5D0
      ISTAR1=ISTART+1
      JEND=ISTAR1-ITOP
      ZMXDT=ZMAX*DTEST
C
      DO 18 J=1,JEND
      INDEX=ISTAR1-J
      ZZZ(INDEX)=ZJ
      FJ=DE-DVZ(INDEX)
      ZJP1=(2.0D0-DH2*FJ)*ZJ-ZJM1
      ZJM1=ZJ
      AZJ=DABS(ZJ)
      IF (AZJ-ZMXDT) 20,20,17
   17 IF (DABS(ZJP1)-AZJ) 18,18,22
   18 ZJ=ZJP1
C
   19 IF (ITOP.EQ.1) GO TO 24
   20 KSTOP=INDEX-1
C
      DO 21 K=1,KSTOP
   21 ZZZ(K)=0.0D0
C
      GO TO 24
   22 JSTART=(J+1)/2
      ZJ=ZZZ(ISTAR1-JSTART)
C
      DO 23 J=JSTART,JEND
      INDEX=ISTAR1-J
      ZZZ(INDEX)=ZJ
      DKZ=DSQRT(DABS(DVZ(INDEX)-DE))
```

```
      ZJ=ZJ*DEXP(-DKZ*DH)
      IF (DABS(ZJ).LT.ZMXDT) GO TO 20
   23 CONTINUE
C
      GO TO 19
C
C     ------------------------------NORMALIZE THE FUNCTION
C
   24 DCOEF=1.0D0/ZMAX
C
      DO 25 K=1,NP1
   25 ZZZ(K)=ZZZ(K)*DCOEF
C
C     ------------------------------COMPUTE THE BOTTOM SOLUTION
C
      IF (.NOT.$BOTPR) RETURN
      EXPON=DEXP(-DHB*DSQRT(DVZB-DE))
      ZZZBOT=ZZZ(NP1)*DRORB
C
      DO 26 J=NP2,NP21
      ZZZ(J)=ZZZBOT
      IF (DABS(ZZZBOT).LT.1.0D-60) GO TO 27
   26 ZZZBOT=ZZZBOT*EXPON
C
      RETURN
C
C
   27 DO 28 I=J,NP21
   28 ZZZ(I)=0.0
C
      RETURN
C
C
   29 IF (DSTART.LT.1.0D-60) GO TO 30
      DSTART=DSTART*1.0D-10
      GO TO 7
   30 WRITE (NPRINT,31)
      J=1
      GO TO 27
C
C
   31 FORMAT ('1 EXPONENT PROTECTION EXCEDED IN SHAPE4',//,
     1       '  RETURNED TO CALLING PROGRAM INCOMPLETE')
      END
```

```
      SUBROUTINE OUTPUT (DUZ,NPTS)
      IMPLICIT REAL*8 (A-H, V-Z), LOGICAL*4 ($)
C
      DIMENSION DZZ(NPTS), DUZ(NPTS), DCC(NPTS)
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< COMMON BLOCKS >>>>>>
      COMMON / D4 / DRO,DRB,DRORB
      COMMON / D5 / CMAX,CMIN,CB,CBSQ
      COMMON / D6 / H,CH,HPLCT,DHHLF,DH2SIX,DH2C3,
     1              DH2,HBOTT,DHB
      COMMON / D7 / DK,DKMIN,DKMAX,DKCLC,DKU,DKL,
     2              DTEST,DSMAX,DE
      COMMON / D9 / ZMAX,CSS,DECAY,DPVEL,DGVEL,GAM
      COMMON / DN / FQV(20),FQ,W,W2
      COMMON /AWKB/ DKN,DINT,DIFF,DS,DM
      COMMON /HEAD/ HED(20)
      COMMON / I1 / NUMV,NMOD,NFREQ,NMODF,N,NP1,
     3              NP21,NPTS,MODE,IT,NP2
      COMMON / IO / NREAD,NPRINT,NPUNCH
      COMMON / L1 / $GRAPH,$BOTPR,$CARDS,$CEL
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< PROGRAM BEGINS >>>>>
C
C     ---------------------------------MODE OUTPUT SECTION
C
    1 WRITE (NPUNCH,6) MODE,FQ,DKN,DPVEL,DGVEL,CSS,DECAY
      WRITE (NPUNCH,7) DUZ
C
      IF (.NOT.$GRAPH) GO TO 2
      WRITE (NPRINT,9) MODE
      RETURN
C
    2 WRITE (NPRINT,10)
      RETURN
C
C     ---------------------------------SOUND PROFILE OUTPUT
C
      ENTRY PROFIL (DZZ, DCC, NPTS)
      WRITE (NPUNCH,3) HED
      WRITE (NPUNCH,4) NPTS,DRO,DRB,CMIN,CB,H,HPLCT
      WRITE (NPUNCH,8) DZZ
      WRITE (NPUNCH,8) DCC
      WRITE (NPRINT,5)
      RETURN
C
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< FORMAT STATEMENTS >>
C
    3 FORMAT (10A8)
    4 FORMAT (I1C,2F10.5,4F10.2)
    5 FORMAT ('0SOUND VELOCITY DETAILED PROFILE PUNCHED')
    6 FORMAT (I5,F5.0,G20.12,2F10.2,2G15.8)
    7 FORMAT (10F8.5)
    8 FORMAT (10F8.1)
    9 FORMAT ('0  OUTPUT FOR MODE', I4,' HAS BEEN PUNCHED')
   10 FORMAT ('+', T118, 'OUTPUT PUNCHED')
      END
```

```
      SUBROUTINE INPUT2
      IMPLICIT REAL*8 (A-H,V-Z), LOGICAL*4 ($)
C
C     ----------------------------------------
C     LAST CHANGED 3 AUGUST 1973
C     ----------------------------------------
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< COMMON BLOCKS >>>>>>
C
      COMMON / D3 / CC(50),CEP(50)
      COMMON / D4 / DRC,DRB,DRORB
      COMMON / D5 / CMAX,CMIN,CB,CBSQ
      COMMON / D6 / H,DH,HPLCT,DHHLF,DH2SIX,DH2C3,
     1              DH2,HBOTT,DHB
      COMMON / D7 / DK,DKMIN,DKMAX,DKCLC,DKU,DKL,
     2              DTEST,DSMAX,DE
      COMMON / DW / FQV(20),FQ,W,W2
      COMMON /HEAD/ HED(20)
      COMMON / I1 / NUMV,NMOD,NFREQ,NMODF,N,NP1,
     3              NP21,NPTS,MODE,IT,NP2
      COMMON / IC / NREAD,NPRINT,NPUNCH
      COMMON / L1 / $GRAPH,$BOTPR,$CARDS,$CEL
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< DATA >>>>>>>>>>>>>>>
C
      DATA CONVRT /1.000000000000000D0/
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< PROGRAM BEGINS >>>>>
C
C
C     -----------------------------------READ IN THE HEADING
C                                        AND RUN PARAMETERS
C
      READ (NREAD,9) HED
      WRITE (NPRINT,13) HED
      READ (NREAD,10) NUMV,NMOD,IEX,N,IOUT,IDEV
C
C     ----------------------------------TEST THE RUN PARAMETERS
C
      IF (N.GT.1000) N=1000
      IF (NUMV.GT.50) NUMV=50
      IF (NMOD.GT.100) NMOD=100
      IEX=IABS(IEX)
      IF (IEX.GT.14) IEX=14
      DTEST=10.0D0**(-IEX)
      IF (IOUT.LE.2) $GRAPH=.TRUE.
      IF ((IOUT.EQ.2).OR.(IOUT.EQ.3)) $CEL=.TRUE.
      IF ((IOUT.EQ.1).OR.(IOUT.EQ.4)) $CARDS=.TRUE.
      IF (IDEV.NE.0) NPUNCH=IDEV
      NP1=N+1
      NP21=N+21
      READ (NREAD,11) H,CB,DRC,DRB,HPLCT
C
C     ----------------------------------READ IN THE FREQUENCIES
C
      READ (NREAD,10) NFREQ
      IF (NFREQ.GT.20) NFREQ=20
      READ (NREAD,11) (FQV(I),I=1,NFREQ)
      FQ=FQV(1)
C
C     -----------------------------------CONVERT AND DISPLAY
C                                        THE RUN PARAMETERS
C
      IF (NUMV) 1,2,2
    1 CONVRT=0.3048
      NUMV=-NUMV
      H=H*CONVRT
      CB=CB*CONVRT
      HPLCT=HPLCT*CONVRT
    2 WRITE (NPRINT,14)
      WRITE (NPRINT,15) (FQV(I),I=1,NFREQ)
      WRITE (NPRINT,16) NMOD,IEX
```

```
      WRITE (NPRINT,17) H,HPLOT
      WRITE (NPRINT,18) CB,N
      WRITE (NPRINT,19)
      IF ($GRAPH) WRITE (NPRINT,22)
      IF ($CARDS) WRITE (NPRINT,21)
      IF ($CEL) WRITE (NPRINT,20) IDEV
      WRITE (NPRINT,23) FED
C
CC    ------------------------------READ IN THE DEPTHS
CC                                   AND SOUND VELOCITIES
CC
C
      DO 3
      READ (NREAD,11) DEPI,CCI
      DEPI=DEPI*CONVRT
      CCI=CCI*CONVRT
      CC(I)=CCI
      DEP(I)=DEPI
      WRITE (NPRINT,12) DEPI,CCI
    3 CONTINUE
C
CC
CC    ---------------------------------COMPUTE SOME CONSTANTS
C
      DH=H/DFLOAT(N)
      DRCRB=DRC/DRB
      DH2=DH*DH
      CBSQ=CB*CB
      DH2SIX=DH2*0.16666666666667
      DH2D3=DH2SIX*2.0D0
      DHHLF=DH*0.5D0
      IF (DEPI-H) 4,6,5
    4 NUMV=NUMV+1
    5 CC(NUMV)=(H-DEPI)*0.0165D0+CCI
      DEP(NUMV)=H
    6 NPTS=NP21
      IF (HPLOT-H) 7,24,8
    7 HPLOT=H
   24 $BOTPR=.FALSE.
      NPTS=NP1
      RETURN
    8 HBOTT=HPLOT-H
      DHB=HBOTT*0.5D-1
      RETURN
C
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< FORMAT STATEMENTS >>
C
    9 FORMAT (10A8)
   10 FORMAT (5I10,I2)
   11 FORMAT (8F10.3)
   12 FORMAT ('0', T25, F6.1, T57, F6.1)
   13 FORMAT ('1', 10A8)
   14 FORMAT ('0', T34, 'RUN PARAMETERS:')
   15 FORMAT ('0', T28, 'FREQUENCIES :',(T42,F6.1,' HZ',/))
   16 FORMAT ('0', T24, 'MODES REQUESTED :', I4, //,
     1    T24,'ITERATION LIMIT : 10**( -', I2, ')')
   17 FORMAT ('0', T27, 'BOTTOM DEPTH :', F8.1,' METERS',//,
     1    T22, 'MAX DEPTH PLOTTED :', F8.1, ' METERS' )
   18 FORMAT ('0', T18, 'BOTTOM SOUND VELOCITY :',F8.1,
     1 ' METERS/SEC',//,T17,'NUMBER OF GRID POINTS :',I5)
   19 FORMAT ('0', T22, 'OUTPUTS REQUESTED : HORIZ WAVE ',
     1 'NUMBER',//,T42,'GROUP AND PHASE VELOCITIES',//)
   20 FORMAT ('0', T42, 'OUTPUT ON FILE FT',I2,'F001')
   21 FORMAT ('0', T42, 'PUNCHED CARD OUTPUT')
   22 FORMAT (' ', T42, 'SOUND VELOCITY AND MODE GRAPHS')
   23 FORMAT ('1', 10A8, //, T33, 'VELOCITY PROFILE',//,T22,
     1    'DEPTH, METERS', T44, 'SOUND VELOCITY, M/SEC', /)
      END
```

```
      SUBROUTINE MODPLT
      IMPLICIT REAL*8 (A-H, V-Z), LOGICAL*4 ($)
      DIMENSION RANGE (4)
C
CCC   *
CCC   *    THIS SUBROUTINE PLOTS THE MODE SHAPE ON THE PRINTE
CCC   *    UTILIZING THE STANDARD (AT NPS) ROUTINE UTPLOT
CCC   *
CCC   *
CCC   ***********************************************************
C
      COMMON / D1 / VELC(1021), DEPTH (1021)
      COMMON / D2 / ZZZ(1021), DVZ(1001)
      COMMON / D5 / CMAX,CMIN,CB,CBSQ
      COMMON / D6 / H,DH,HPLOT,DHHLF,DH2SIX,DH2D3,
     1              DH2,HBOTT,DHB
      COMMON / D7 / DK,DKMIN,DKMAX,DKCLC,DKU,DKL,
     2              DTEST,DSMAX,DE
      COMMON / D8 / DIFFI
      COMMON / D9 / ZMAX,DSS,DECAY,DPVEL,DGVEL,GAM
      COMMON/DKMAP/ DK1(100)
      COMMON / DW / FQV(20),FQ,W,W2
      COMMON /AWKB/ DKN,DINT,DIFF,DS,DM
      COMMON /HEAD/ HED(20)
      COMMON / I1 / NUMV,NMOD,NFREQ,NMODF,N,NP1,
     3              NP21,NPTS,MODE,IT,NP2
      COMMON / IO / NREAD,NPRINT,NPUNCH
C
      WRITE (NPRINT,1) HED,FQ
      WRITE (NPRINT,2) MODE,DK
      WRITE (NPRINT,3)
      CALL UTPLOT (ZZZ,DEPTH,NPTS,RANGE,2,0,PBCTT)
      WRITE (NPRINT,4) DPVEL,DSS,DECAY,DGVEL,IT,DIFFI
      RETURN
C
C
      ENTRY DZPLOT
      PBCTT=SNGL(H)
      WRITE (NPRINT,5) HED
      WRITE (NPRINT,6)
      WRITE (NPRINT,7)
      RANGE(1)=SNGL(CB)
      RANGE(2)=SNGL(CMIN)
      RANGE(4)=0.0
      RANGE(3)=SNGL(HPLOT)
      CALL UTPLOT (VELC,DEPTH,NPTS,RANGE,2,0,PBCTT)
      RANGE(1)=1.0
      RANGE(2)=-1.0
      RETURN
C
C
    1 FORMAT ('1', 10A8, '  FREQUENCY : ', F6.1, ' HZ')
    2 FORMAT ('0', T25, 'MODE', I4, '   :  K =', F16.12)
    3 FORMAT ('0', //, T23, 'NORMALIZED EIGENFUNCTION',
     1  ' VS DEPTH ')
    4 FORMAT ('0PHASE VELOCITY :', F7.1, ' M/SEC.   DSS',
     *                '    :', G15.7,
     1     4X, ' BOTTOM LEAKAGE COEF :', G15.7, //,
     $           ' GROUP VELOCITY :',
     2     F7.1, ' M/SEC.   ITERATIONS :', I5, 1OX,
     3     'DIFFERENCE FUNCTION :', G16.8 )
    5 FORMAT ('1', 10A8)
    6 FORMAT ('0', //, T24, 'GRAPH OF SOUND VELOCITY',
     1           ' VS DEPTH')
    7 FORMAT ('0', T19, 'DEPTH IN METERS.',
     1       ' SOUND VELOCITY IN METERS/SEC.')
      END
```

```
      SUBROUTINE LINEAR
      IMPLICIT REAL*8 (A-H, V-Z), LOGICAL*4 ($)
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< COMMON BLOCKS >>>>>>
C
      COMMON / D1 / VELO(1021), DEPTH (1021)
      COMMON / D2 / ZZZ(1C21), DVZ(1001)
      COMMON / D3 / CC(5C),DEP(50)
      COMMON / D5 / CMAX,CMIN,CB,CBSQ
      COMMON / D6 / H,DH,HPLOT,DHHLF,DH2SIX,DH2D3,
     1              DH2,HBOTT,DHB
      COMMON / D9 / ZMAX,DSS,DECAY,DPVEL,DGVEL,GAM
      COMMON / DW / FQV(20),FQ,W,W2
      COMMON / I1 / NUMV,NMOD,NFREQ,NMODF,N,NP1,
     3              NP21,NPTS,MODE,IT,NP2
      COMMON / L1 / $GRAPH,$BOTPR,$CARDS,$CEL
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< PROGRAM BEGINS >>>>>
C
      CMAX=C.CDO
      CMIN=CB
      DU=0.0D0
      DL=DEP(2)
      VU=CC(1)
      VL=CC(2)
      DLMDU=DL-DU
      VLMVU=VL-VU
      I=2
C
C     ------------------------------------DEPTH LOOP BEGINS
C
C
      DO 4 J=1,NP1
      DEPJ=DH*DFLOAT(J-1)
      DEPTH(J)=DEPJ
      IF (DEPJ-DL) 2,2,1
    1 I=I+1
      DU=DL
      VU=VL
      DL=DEP(I)
      VL=CC(I)
      DLMDU=DL-DU
      VLMVU=VL-VU
C
    2 VELOJ=(DEPJ-DU)*VLMVU/DLMDU+VU
C
      IF (VELOJ.GT.CMAX) CMAX=VELOJ
      IF (VELOJ-CMIN) 3,4,4
C
    3 CMIN=VELOJ
    4 VELO(J)=VELOJ
C
C
C     ------------------------------------DEPTH LOOP ENDS
C
      IF (.NOT.$BOTPR) GO TO 6
      NP2=NP1+1
C
C
C     ------------------------ASSIGN BOTTOM VELOCITY VALUES
C
C
      DO 5 J=1,2C
      VELO(NP1+J)=CB
      DEPTH(NP1+J)=H+DHB*DFLOAT(J-1)
    5 CONTINUE
C
C
C
    6 RETURN
      END
```

```
      SUBROUTINE WKB
      IMPLICIT REAL*8(A-H,V-Z),LOGICAL*4($)
C
CCC   ---------------------------------
CCC   LAST CHANGED 2  7 SEPT 73   AIRY PHASE B.C.
CCC   ---------------------------------
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< COMMON BLOCKS >>>>>>
C
      COMMON / C2 / ZZZ(1021), DVZ(1001)
      COMMON / D4 / DRO,DRB,DRORB
      COMMON / D6 / H,DH,HPLOT,DHHLF,DH2SIX,DH2D3,
     1              DH2,HBOTT,DHB
      COMMON / D7 / DK,DKMIN,DKMAX,DKOLD,DKU,DKL,
     2              DTEST,DSMAX,DE
      COMMON / D10/ WOCB,WCONP1,WCONMN,DVZB
      COMMON /AWKB/ DKN,DINT,DIFF,DS,DM
      COMMON / I1 / NUMV,NMOD,NFREQ,NMODF,N,NP1,
     3              NP21,NPTS,MODE,IT,NP2
      COMMON / I2 / LOMODE,IHMODE,KWELL,IWELL,NWELL,
     4              JHI,JLOW
      COMMON / I3 / JUPPER,JBOTT,ISHAP,JTPL,JTPU
      COMMON / L1 / $GRAPH,$BOTPR,$CARDS,$CEL
      COMMON / P1 / DPI,DPIC2,DPIC4,DPIRT2,D2PI
      COMMON / DTH/ DTHETL
      DATA DEXPU/1.7D2/,DWELL/6.0/
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< PROGRAM BEGINS >>>>>
C
      DINT=0.0D0
      $TOP=.FALSE.
      JTPL=1
      DE=(DKMAX-DKN)*(DKMAX+DKN)
      DKB=DSQRT(DVZB-DE)
      IWELL=1
      ISIG=-1
      DARG=0.0D0
      DARGP=0.0D0
C
CCC   ---------------------------------DEPTH LOOP BEGINS
CCC
C
      DO 21 J=1,NP1
      DKZ2=DE-DVZ(J)
C
CCC   ---------------------------------CHECK FOR CONDITIONS
CCC                                     AND TYPE OF SOLUTION
C
      IF (DKZ2) 1,2,3
    1 IF (ISIG) 4,5,6
    2 IF (ISIG) 7,21,9
    3 IF (ISIG) 10,19,20
C
CCC   ---------------------------------IMAGINARY REGION
C
    4 DARGP=DSQRT(-DKZ2)
      DARG=DARG+DARGP
      GO TO 21
C
    5 ISIG=-1
```

```
      GO TO 4
C
    6 DARGP=DSQRT(-DKZ2)
      DX=DKZ/(DKZ+DARGP)
      DINT=DINT+DKZ*(DX-1.0D0)*0.5D0*DH
      ISIG=-1
      JTPL=J-1
      DARG=(2.0D0-DX)*DARGP*0.5D0
      GO TO 21
C
CC    ----------------------------ZERO VERTICAL WAVE NUMBER
C
    7 IBACK=-1
    8 ISIG=0
      GO TO 21
C
    9 IBACK=1
      GO TO 8
C
CC    ----------------------------------REAL REGION
C
   10 ISIG=1
      IF ($TOP) GO TO 11
      JTPU=J
   11 DKZ=DSQRT(DKZ2)
      DX=DARGP/(DKZ+DARGP)
      DARG=DARG+DARGP*(DX-1.0D0)*0.5D0
      IF ($TOP) GO TO 13
   12 D2=2.0D0*DEXP(2.0D0*DARG*DH)
      DTHETU=DATAN((D2-1.0D0)/(D2+1.0D0))
      DINT=DTHETU
      $TOP=.TRUE.
      GO TO 18
   13 DT1=DMOD(DINT,D2PI)
      IF (DARG*DH-DWELL) 15,14,14
   14 IF (IWELL.EQ.KWELL) GO TO 28
      DARG=0.0D0
      IWELL=IWELL+1
      JTPU=J
      DTHETU=DPI04
      DINT=DTHETU
      GO TO 18
   15 A=DSIN(DT1)
      B=DCOS(DT1)
      DEL=DEXP(DARG*DH)
      DEML=DEXP(-DARG*DH)
      AC=DEL*(A+B)+DEML*(A-B)
      BC=DEL*(A+B)-DEML*(A-B)
      DINTP=DINT-DT1+DATAN2(AC,BC)
   16 IF (DINTP.GE.DINT-DTEST) GO TO 17
      DINTP=DINTP+DPI02
      GO TO 16
   17 DINT=DINTP
   18 DINT=DINT+(2.0D0-DX)*DKZ*0.5D0*DH
      DARG=0.0D0
      GO TO 21
C
   19 DKZ=DSQRT(DKZ2)
      ISIG=+1
      DINT=DINT+DKZ*DH
      IF ($TOP) GO TO 13
      JTPU=J
      GO TO 12
C
   20 DKZ=DSQRT(DKZ2)
      DINT=DINT+DKZ*DH
C
   21 CONTINUE
C
CC
CC    ----------------------------------DEPTH LOOP ENDS
C
```

```
      IF (DKZ2) 22,27,25
C
C     ---------------------------------DECAYING SOLUTION
C
   22 DKS=DARGP
      DARG=(DARG-0.5D0*DARGP)*DH*2.0D0
      IF (DARG-DEXPU) 23,24,24
   23 D3=(DKS-DRORB*DKB)*DEXP(-DARG)*2.0D0/(DKS+DRORB*DKB)
      DTHETL=DATAN((1.0D0+D3)/(1.0D0-D3))
      GO TO 27
   24 DTHETL=DPI04
      GO TO 27
C
C     ---------------------------------BOTTOM REFLECTED
C
   25 JTFL=NP1
      IF (DKB.EQ.0.0D0) GO TO 26
      DTHETL=DATAN(DKZ/(DKB*DRORB))
      GO TO 27
C
   26 DTHETL=DPI02
C
C     ---------------------------------COMPUTE DIFFERENCE
C
   27 DS=DINT+DTHETL
      DINT=DINT-DTHETU
      DIFF=DS-DM*DPI
      NWELL=IWELL
      RETURN
C
   28 IWELL=IWELL+1
      GO TO 24
C
      END
```

BIBLIOGRAPHY

Arthur D. Little, Inc., Report No. C-70673, Design of an Array to Excite Individual Normal Modes in the BIFI Range, 31 January 1969.

Biot, M. A., "General Theorems on Equivalence of Group Velocity and Energy Transport," The Physical Review, vol. 105, 1129, 1957.

Bucker, H. P. and Morris, H. E., "Normal Mode Intensity Calculations for a Constant-Depth Shallow-Water Channel," J. Acoustic Soc. Amer., vol. 38, p. 1010, 1963.

Bucker, H. P. and Morris, H. E., "Epstein Normal-Mode Model of a Surface Duct," J. Acoustic Soc. Amer., vol. 41, p. 1475, 1967.

Churchill, R. V., Fourier Series and Boundary Value Problems, McGraw-Hill, 1941.

Naval Underwater Systems Center Technical Report 4319, Computer Programs to Calculate Normal Mode Propagation and Applications to the Analysis of Explosive Sound Data in the BIFI Range, by W. G. Kanabis, 6 November 1972.

Kornhauser, E. T., and Raney, W. P., "Attenuation in Shallow Water Propagation Due to an Absorbing Bottom," J. Acoustic Soc. Amer., vol. 27, p. 689, 1955.

North Atlantic Treaty Organization, SACLANT ASW Research Centre Technical Report No. 203, An Expansion Technique for Rendering the WKBJ Method Uniformly Valid, by V. R. Lauvstad, 15 December 1971.

Naval Research Laboratory Memorandum Report 2381, A Normal Mode Computer Program for Calculating Sound Propagation in Shallow Water with an Arbitrary Velocity Profile, by A. V. Newman and F. Ingenito, January 1972.

Officer, C. B., Introduction to the Theory of Sound Transmission, McGraw-Hill, 1958.

Schiff, L. I., Quantum Mechanics, McGraw-Hill, 1955.

Tolstoy, I. and Clay, C. S., Ocean Acoustics, McGraw-Hill, 1966.

Williams, A. O., Jr., and Horne, W., "Axial Focusing of Sound in the Sofar Channel," J. Acoustic Soc. Amer., vol. 41, p. 189, 1967.

Williams, A. O., "Normal Mode Methods in the Propagation of Underwater Sound," Underwater Acoustics, ed. by R. W. B. Stephens, Wiley-Interscience, 1970.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Professor Glenn H. Jung, Code 58Jg Department of Oceanography Naval Postgraduate School Monterey, California 93940	3
4.	Associate Professor Alan B. Coppens, Code 61Cz Department of Physics and Chemistry Naval Postgraduate School Monterey, California 93940	3
5.	Department of Oceanography Naval Postgraduate School Monterey, California 93940	3
6.	The Oceanographer of the Navy The Madison Building 732 North Washington Street Alexandria, Virginia 22314	1
7.	LT. Kirk E. Evans, USN SMC 2886 Naval Postgraduate School Monterey, California 93940	3
8.	Dr. Robert E. Stevenson Scientific Liaison Office Scripps Institute of Oceanography La Jolla, California 92037	1
9.	Office of Naval Research, Code 480 Arlington, Virginia	1
10.	NAVOCEANO Library U.S. Naval Oceanographic Office Washington, D. C. 20390	1

No. Copies

11. LCDR Ernest Young — 2
Office of Naval Research Liaison Officer
Room 411, Herrmann Hall East Wing
Naval Postgraduate School
Monterey, California 93940

12. Associate Professor Warren W. Denner, Code 58Dw — 1
Department of Oceanography
Naval Postgraduate School
Monterey, California 93940

13. Assistant Professor Robert H. Bourke, Code 58Bf — 1
Department of Oceanography
Naval Postgraduate School
Monterey, California 93940

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
Two Methods for the Numerical Calculation of Acoustic Normal Modes in the Ocean

5. TYPE OF REPORT & PERIOD COVERED
Master's Thesis; September 1973

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)
Kirk Eden Evans

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Naval Postgraduate School
Monterey, California 93940

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS
Naval Postgraduate School
Monterey, California 93940

12. REPORT DATE
September 1973

13. NUMBER OF PAGES
161

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)
Naval Postgraduate School
Monterey, California 93940

15. SECURITY CLASS. (of this report)
Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)
Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Normal mode	WKB	Underwater sound
Acoustics	Ocean acoustics	
Sound propagation	SOFAR	
Wave equation	Deep sound channel	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
Three computer programs were written to find the eigenvalues and eigenfunctions of acoustic normal modes in the ocean. The programs used two different methods: an iterative finite difference scheme, and a method based upon the WKB approximation of quantum mechanics. The methods assume a flat fluid bottom and are designed for any arbitrary sound speed profile. While the results of both the finite difference and the WKB methods agreed, the WKB method proved faster.